CAPÍTULO I

MANUAL TÉCNICO

1.1 INTRODUCCIÓN

La finalidad de este documento, es explicar las distintas tecnologías involucradas en el desarrollo de nuestro proyecto web. Las Herramientas utilizadas son Open Source.

Fue necesario la instalación de un Servidor de Aplicaciones Web, en nuestro sitio utilizamos Wamp Server 2.1a, que realiza la instalación completa de los servicios de Apache, Php y Mysql como motor de Base de datos. Ya es de nuestra responsabilidad, el distinto administrador de base a utilizar.

Las versiones de los servicios son:

- Apache 2.2.17
- Php 5.5.3
- MySql 5.5.8
- PhpMyAdmin 3.2.0.1

Existen algunos CMS para la creación de nuestro proyecto, nuestra elección fue Joomla, versión 1.5.9, la justificación de la versión es por considerarse un producto estable. Los distintos componentes utilizados para el proyecto web fueron bajados del sitio oficial (www.joomla.org).

El uso de herramientas Php, JavaScript, CSS, Html, en conjunto nos permite elaborar un producto con Tecnología Ajax, haciendo que el dinamismo en la navegación esté a la orden del día y la interacción Hombre-Máquina sea una experiencia super agradable.

1.2 EL FRAMEWORK

Nuestro proyecto no vio la necesidad de utilizar un Framework profesional, ya que Joomla, posibilita la adaptación necesaria de los recursos de un Mashup de contenidos. Aunque existen en el mercado Herramientas de soporte, tanto para facilitar el diseño de la web, o para contruir la Lógica del Negocio. Para nuestro caso solo fue necesario buscar o comprar las extensiones de Joomla con una funcionalidad específica. El resultado final, lo observamos en un navegador.

1.3 ESTRUCTURA

La estructura del sitio no es tan fácil descubrirla, ya que Joomla nos facilita la construcción de formularios, donde se incluyen opciones, aquellas opciones crean las tablas necesarias en bases de datos, funciones y clases php, estas automáticamente engranan entre sí, para obedecer el comportamiento esperado por el usuario.

Aclaramos que no solo creamos formularios, sino implementamos funcionalidades necesarias en un Mashup de noticias, Ejemplo: Chat directo, Noticias

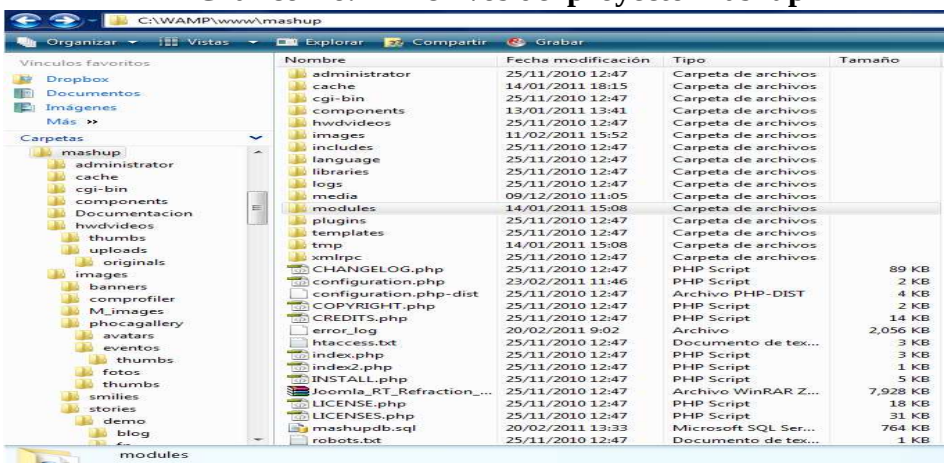
en Vivo, Blog de Noticias, Clasificados, etc. Esas funcionalidades al igual que los formularios, de igual manera construyen las funciones y tablas necesarias para el funcionamiento esperado. Esto es completamente transparente al usuario, es esa la finalidad de los CMS, facilidad y cero complicaciones con la programación avanzada.

Joomla, para su organización, gestiona:

- Secciones
- Categorías
- Artículos
- Usuarios
- Módulos
- Plantillas
- Idiomas
- Menús
- Archivos Multimedia

Como todo sitio web, el proyecto se aloja en la carpeta raíz www del servidor:

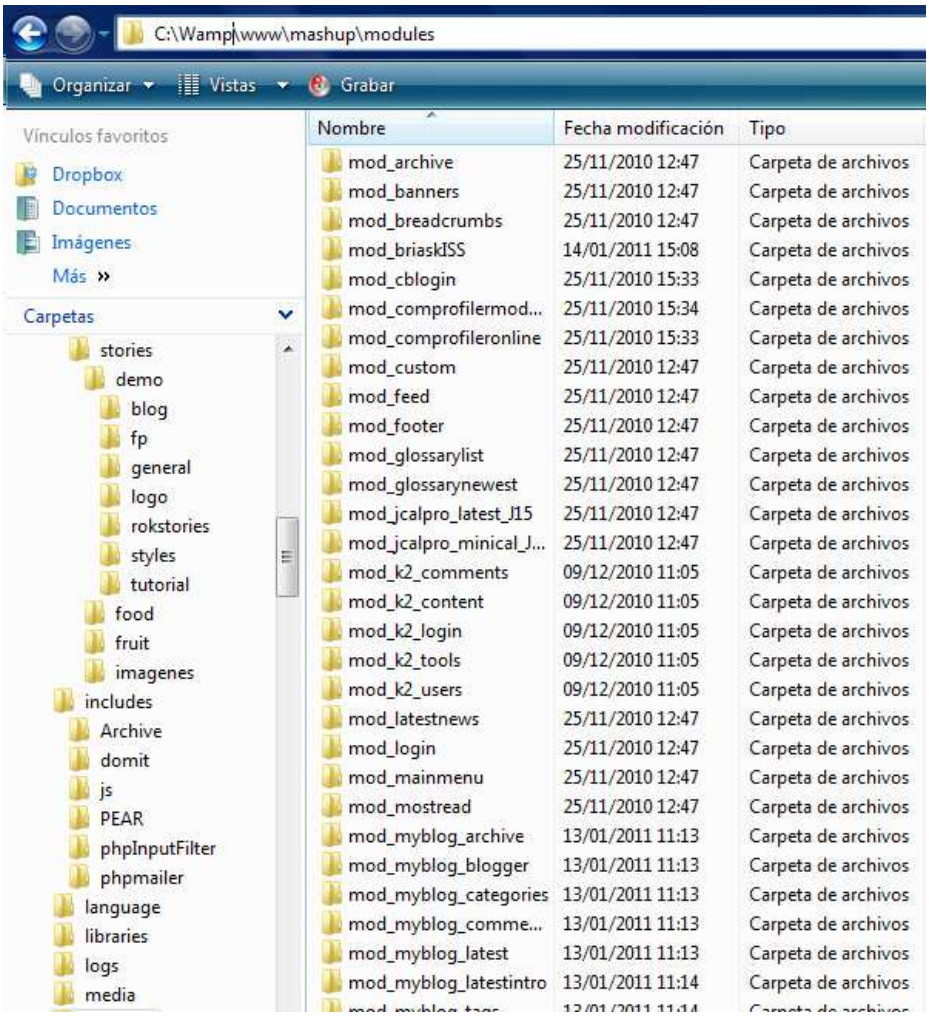
Gráfico No. 1 Archivos del proyecto Mashup



Elaborado por: Araceli Dávila M.

La carpeta módulos, contiene los archivos que participan en cada componente, es decir los archivos .php:

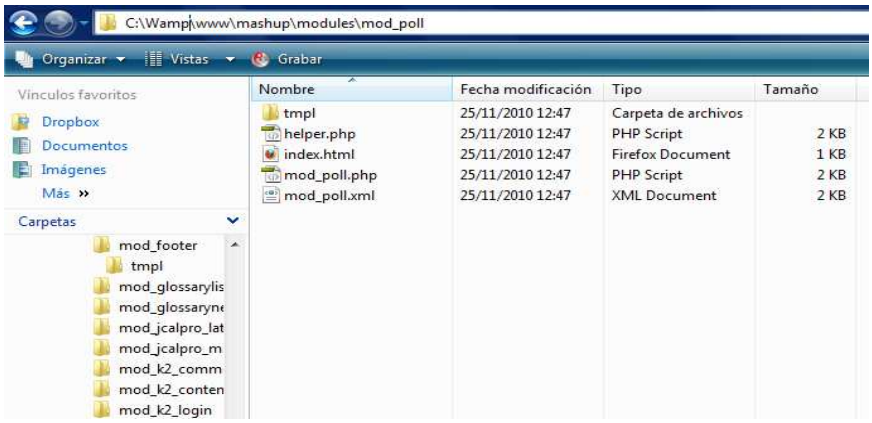
Gráfico No. 2 Módulos participantes del Mashup



Elaborado por: Araceli Dávila M.

Para dar un ejemplo, observaremos los archivos del módulo de encuestas (Polls), estos son los archivos:

Gráfico No. 3 Archivos pertenecientes al módulo de encuesta



Elaborado por: Araceli Dávila M.

Si vemos la programación del archivo mod_poll.php, observaremos las funciones y variables participantes en la funcionalidad solicitada por el usuario.

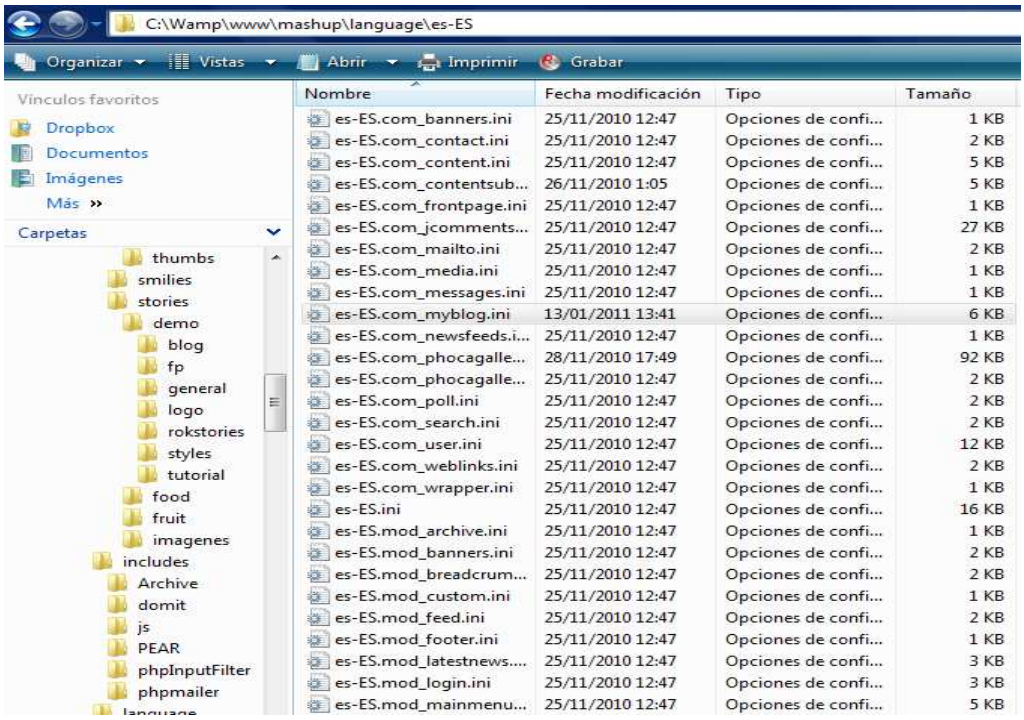
Gráfico No. 4 Archivo mod_poll.php

```php
<?php
/**
* @version      $Id: mod_poll.php 10381 2008-06-01 03:35:53Z pasamio $
* @package      Joomla
* @copyright    Copyright (C) 2005 - 2008 Open Source Matters. All rights reserved.
* @license      GNU/GPL, see LICENSE.php
* Joomla! is free software. This version may have been modified pursuant
* to the GNU General Public License, and as distributed it includes or
* is derivative of works licensed under the GNU General Public License or
* other free or open source software licenses.
* See COPYRIGHT.php for copyright notices and details.
*/

// no direct access
defined('_JEXEC') or die('Restricted access');

// Include the syndicate functions only once
require_once (dirname(__FILE__).DS.'helper.php');

$tabclass_arr = array ('sectiontableentry2', 'sectiontableentry1');

$menu    = &JSite::getMenu();
$items   = $menu->getItems('link', 'index.php?option=com_poll&view=poll');
$itemid = isset($items[0]) ? '&Itemid='.$items[0]->id : '';

$poll    = modPollHelper::getPoll($params->get( 'id', 0 ));

if ( $poll && $poll->id ) {
    $layout = JModuleHelper::getLayoutPath('mod_poll');
    $tabcnt = 0;
    $options = modPollHelper::getPollOptions($poll->id);

    require($layout);
}
```

Elaborado por: Araceli Dávila M.

El resto de la funcionalidad, se incluye en las variables, asistidas en la siguiente sección:

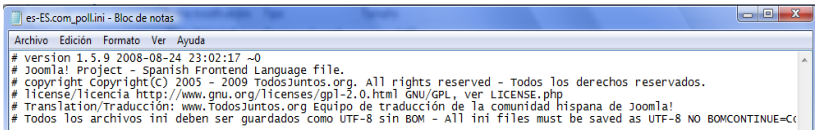
Gráfico No. 5 Archivos de Inicialización de módulos por idioma


C:\Wamp\www\mashup\language\es-ES
Organizar
Vistas
Abrir
Imprimir
Grabar
Vínculos favoritos
Dropbox
Documentos
Imágenes
Más
Carpetas
thumbs
smilies
stories
demo
blog
fp
general
logo
rokstories
styles
tutorial
food
fruit
imagenes
includes
Archive
domit
js
PEAR
phpInputFilter
phpmailer
Nombre
Fecha modificación
Tipo
Tamaño
es-ES.com_banners.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.com_contact.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.com_content.ini 25/11/2010 12:47 Opciones de confi... 5 KB
es-ES.com_contentsub... 26/11/2010 1:05 Opciones de confi... 5 KB
es-ES.com_frontpage.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.com_jcomments... 25/11/2010 12:47 Opciones de confi... 27 KB
es-ES.com_mailto.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.com_media.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.com_messages.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.com_myblog.ini 13/01/2011 13:41 Opciones de confi... 6 KB
es-ES.com_newsfeeds.i... 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.com_phocagalle... 28/11/2010 17:49 Opciones de confi... 92 KB
es-ES.com_phocagalle... 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.com_poll.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.com_search.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.com_user.ini 25/11/2010 12:47 Opciones de confi... 12 KB
es-ES.com_weblinks.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.com_wrapper.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.ini 25/11/2010 12:47 Opciones de confi... 16 KB
es-ES.mod_archive.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.mod_banners.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.mod_breadcrum... 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.mod_custom.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.mod_feed.ini 25/11/2010 12:47 Opciones de confi... 2 KB
es-ES.mod_footer.ini 25/11/2010 12:47 Opciones de confi... 1 KB
es-ES.mod_latestnews.... 25/11/2010 12:47 Opciones de confi... 3 KB
es-ES.mod_login.ini 25/11/2010 12:47 Opciones de confi... 3 KB
es-ES.mod_mainmenu... 25/11/2010 12:47 Opciones de confi... 5 KB

Elaborado por: Araceli Dávila M.

El contenido del archivo es-ES.com_poll.ini, contiene las variables globales de la funcionalidad del módulo:

Gráfico No. 6 Contenido de las Variables globales en el archivo es-ES.com_poll.ini


es-ES.com_poll.ini - Bloc de notas
Archivo Edición Formato Ver Ayuda
version 1.5.9 2008-08-24 23:02:17 -0
Joomla! Project - Spanish Frontend Language file.
copyright Copyright(C) 2005 - 2009 TodosJuntos.org. All rights reserved - Todos los derechos reservados.
license/licencia http://www.gnu.org/licenses/gpl-2.0.html GNU/GPL, ver LICENSE.php
Translation/Traducción: www.TodosJuntos.org Equipo de traducción de la comunidad hispana de Joomla!
Todos los archivos ini deben ser guardados como UTF-8 sin BOM - All ini files must be saved as UTF-8 NO BOMCONTINUE=C

Elaborado por: Araceli Dávila M.

1.4 DATOS TÉCNICOS DEL SITIO

Dominio: Adquirido a la compañía Hostgator(www.hostgator.com)

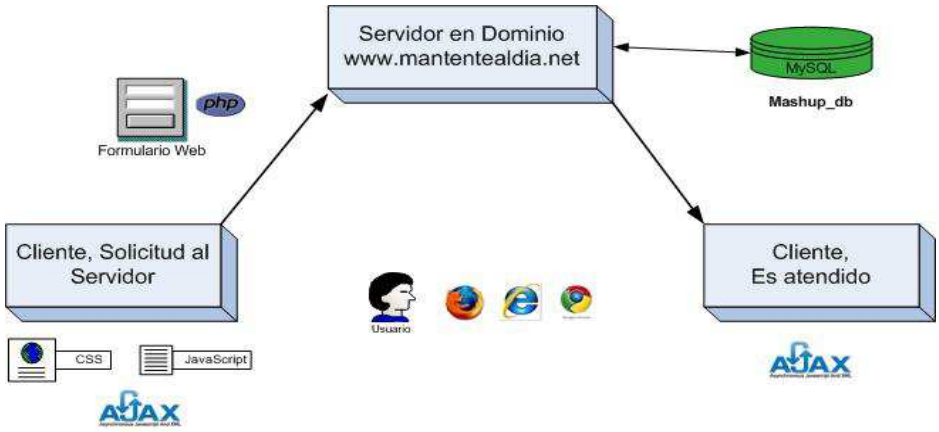
Hosting: Adquirido a la compañía Hostgator(www.hostgator.com)

Características Generales:

Espacio en Disco	800 MB
Banda Ancha	5 GIGS
Dominios Compartidos	1
Manejador de Archivos	Si
Soporte 24x7	Si
Backups Programados	Si
Mysql Bases	1
Cuentas FTP	1
Php5	Compatible
FTP	Si
Cuentas Pop3	Ilimitados
WebMail	Si
Panel de Control	Si
phpMyAdmin	Si
Logs de Errores	Si
Dirección IP Real	Si
Máximo de Conexiones Simultaneas a base de datos	30 usuarios concurrentes

1.5 DIAGRAMA DE COLABORACIÓN

Gráfico No. 7 Diagrama de Colaboración



Elaborador por: Araceli Dávila M.

1.6 DIAGRAMA DE FLUJO DE DATOS

1.6.1 Diagrama de Contexto

Gráfico Mo. 8 Diagrama de Contexto o Nivel cero.

Elaborado por: Araceli Dávila M.

1.6.2 Diagrama de Nivel Superior

Diagrama No. 9 Diagrama de Nivel Superior o Nivel uno

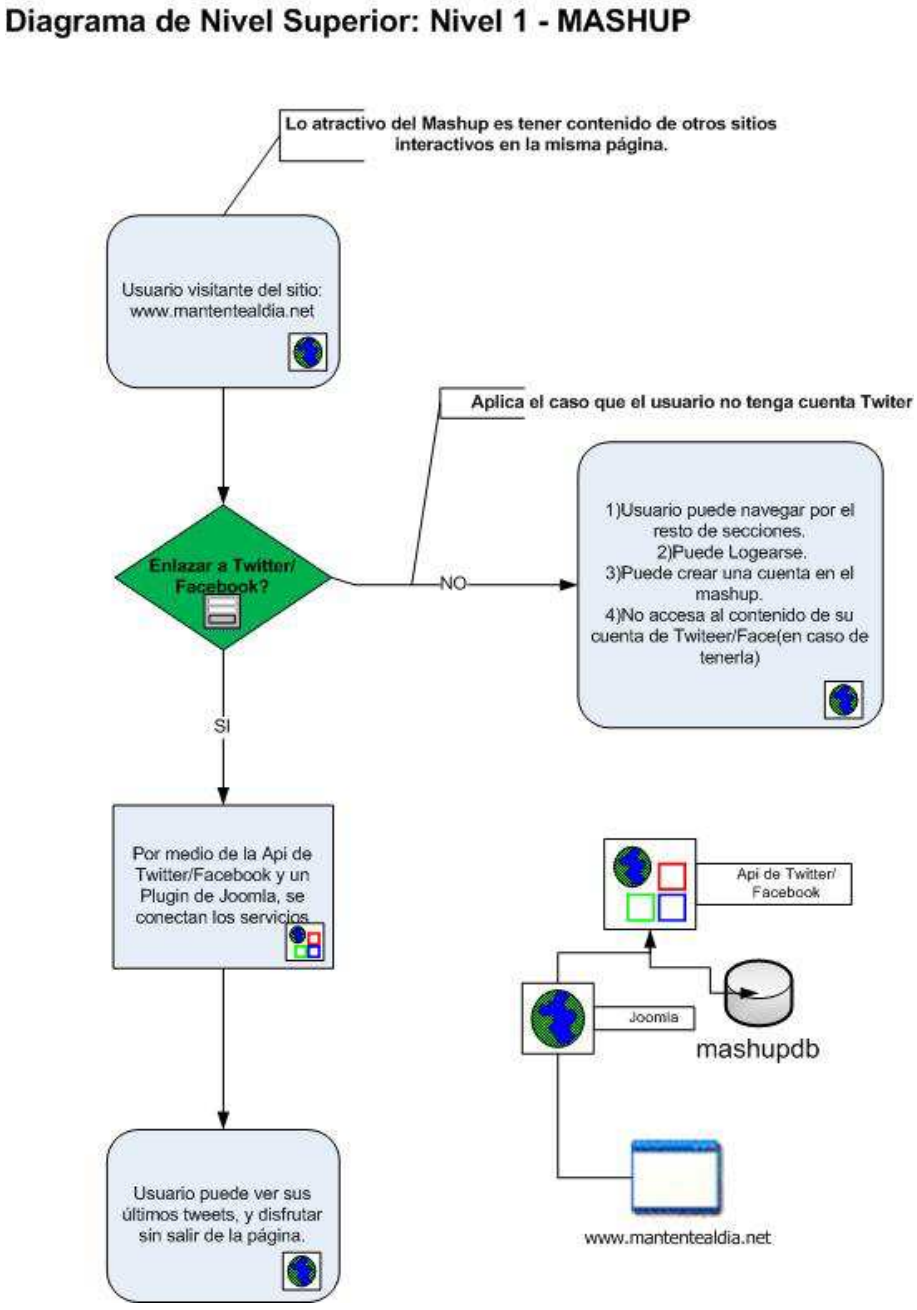


Diagrama de Nivel Superior: Nivel 1 - MASHUP
Lo atractivo del Mashup es tener contenido de otros sitios interactivos en la misma página.
Usuario visitante del sitio: www.mantentealdia.net
Aplica el caso que el usuario no tenga cuenta Twiter
Enlazar a Twitter/ Facebook?
NO
1)Usuario puede navegar por el resto de secciones.
2)Puede Logearse.
3)Puede crear una cuenta en el mashup.
4)No accesa al contenido de su cuenta de Twiteer/Face(en caso de tenerla)
SI
Por medio de la Api de Twitter/Facebook y un Plugin de Joomla, se conectan los servicios
Api de Twitter/ Facebook
Joomla
mashupdb
Usuario puede ver sus últimos tweets, y disfrutar sin salir de la página.
www.mantentealdia.net

Elaborado por: Araceli Dávila M.

1.6.3 Diagrama de Nivel Inferior

Diagrama No.10 Diagrama de Nivel Inferior o Nivel 2

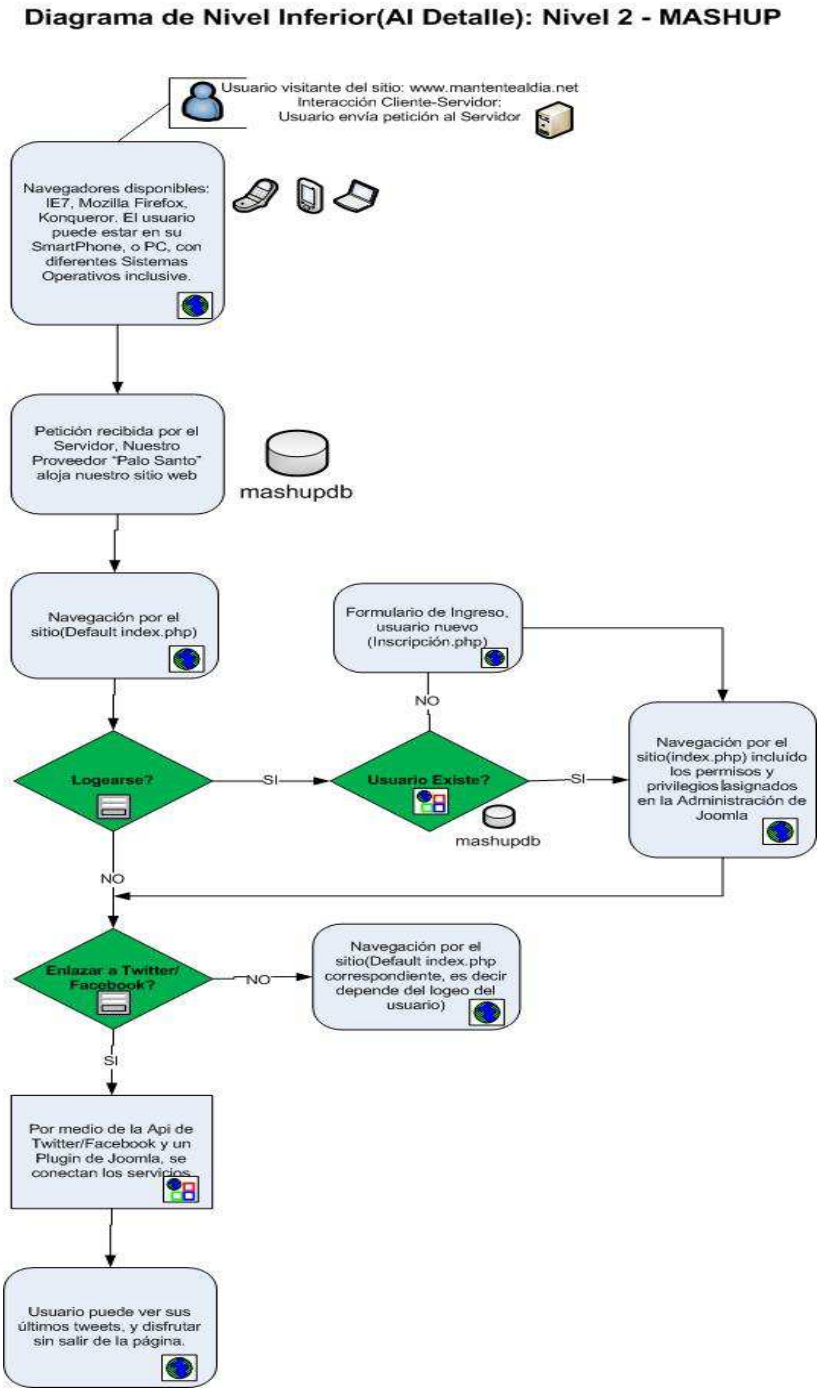
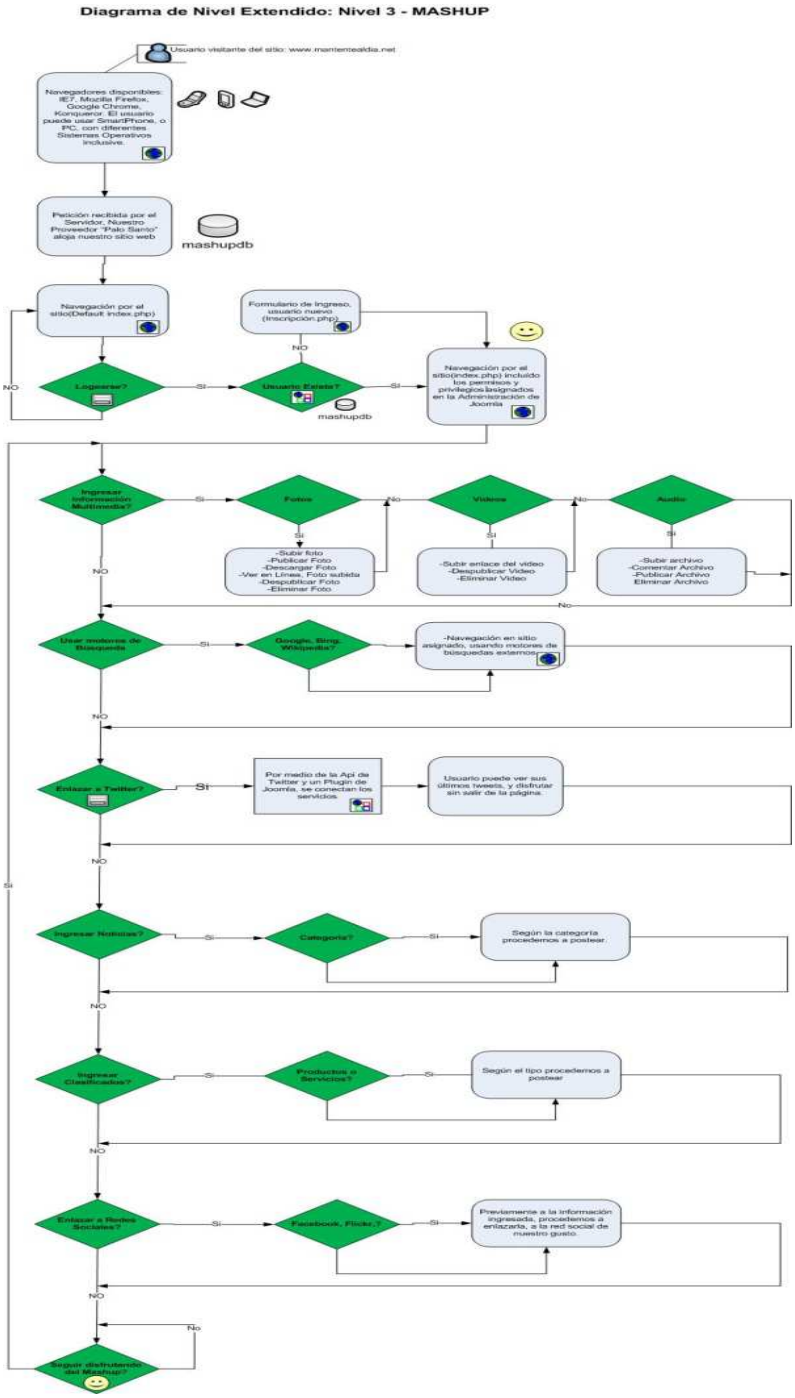


Diagrama de Nivel Inferior(Al Detalle): Nivel 2 - MASHUP
Usuario visitante del sitio: www.mantentealdia.net
Interacción Cliente-Servidor:
Usuario envía petición al Servidor
Navegadores disponibles: IE7, Mozilla Firefox, Konqueror. El usuario puede estar en su SmartPhone, o PC, con diferentes Sistemas Operativos inclusive.
Petición recibida por el Servidor, Nuestro Proveedor "Palo Santo" aloja nuestro sitio web
mashupdb
Navegación por el sitio(Default index.php)
Formulario de Ingreso, usuario nuevo (Inscripción.php)
NO
Logearse?
SI
Usuario Existe?
SI
mashupdb
Navegación por el sitio(index.php) incluído los permisos y privilegios lasignados en la Administración de Joomla
NO
Enlazar a Twitter/ Facebook?
NO
Navegación por el sitio(Default index.php correspondiente, es decir depende del logeo del usuario)
SI
Por medio de la Api de Twitter/Facebook y un Plugin de Joomla, se conectan los servicios
Usuario puede ver sus últimos tweets, y disfrutar sin salir de la página.

Elaborador por: Araceli Dávila M.

1.6.4 Diagrama de Nivel Extendido

Diagrama No. 11 Diagrama de Niveles extendidos o Nivel 3



Elaborado por: Araceli Dávila M.

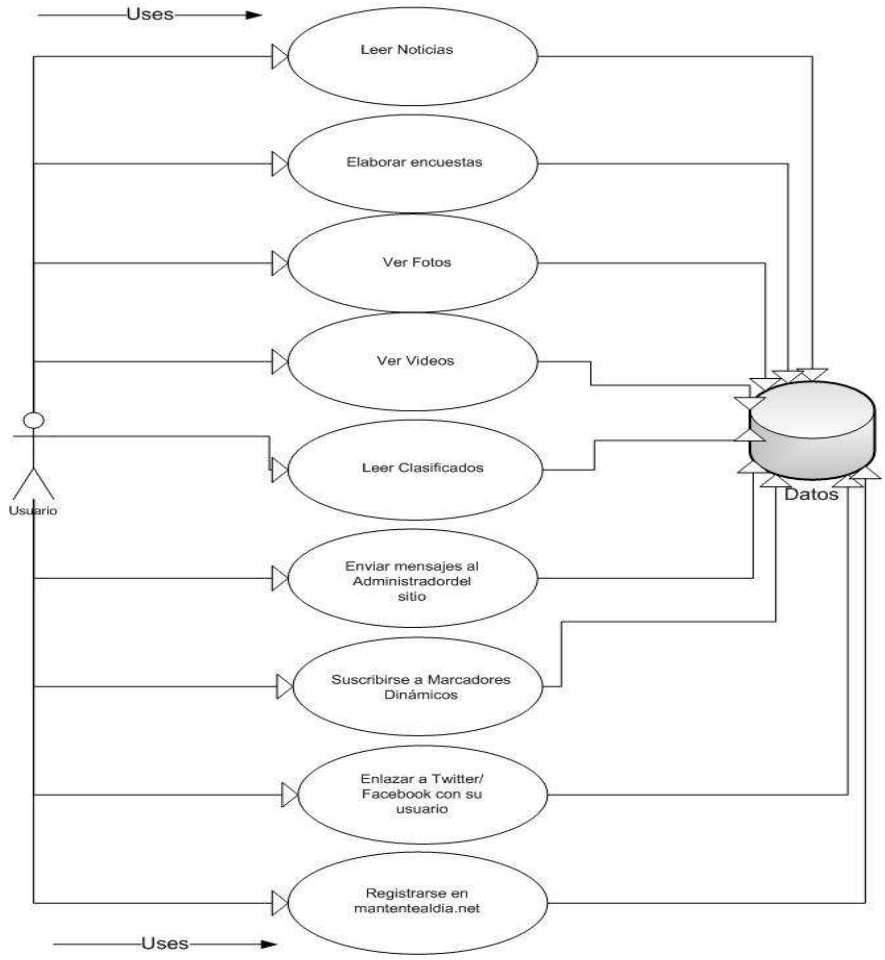
1.7 DIAGRAMA DE CASOS DE USO

Sirve como una técnica para la captura de requisitos potenciales de un nuevo sistema o actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

Caso de Uso:

1er Nivel:

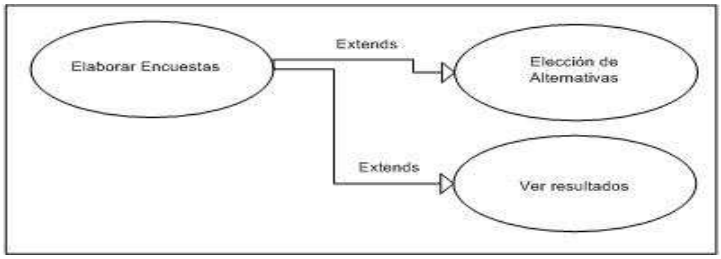
Gráfico No. 12 Casos de Uso Primer Nivel



Elaborador por: Araceli Dávila M.

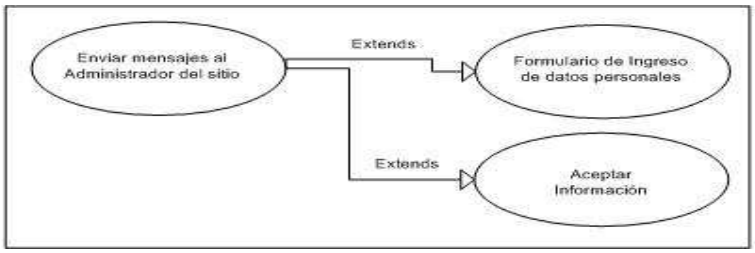
Caso de Uso 2do. Nivel

Gráfico No. 13 Casos de Uso Nivel II – Elaboración de Encuestas



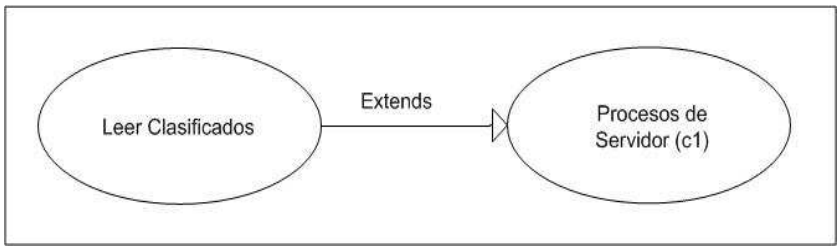
Elaborador por: Araceli Dávila M.

Gráfico No. 14 Casos de Uso Nivel II – Envío de Mensajes al Administrador del Sitio



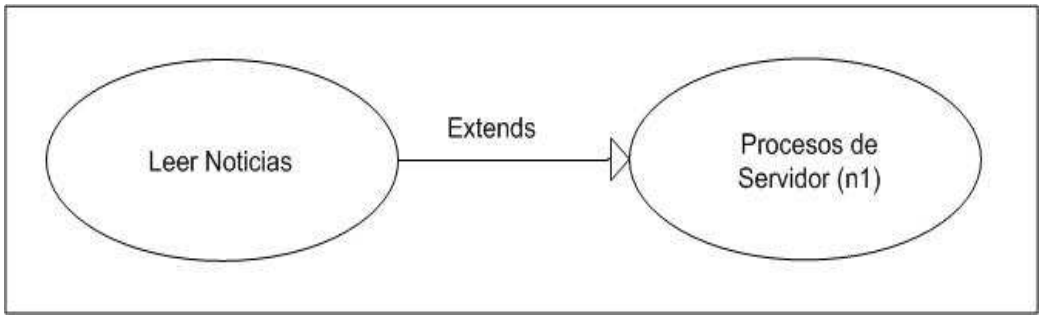
Elaborador por: Araceli Dávila M.

Gráfico No. 15 Casos de Uso Nivel II – Lectura de Clasificados



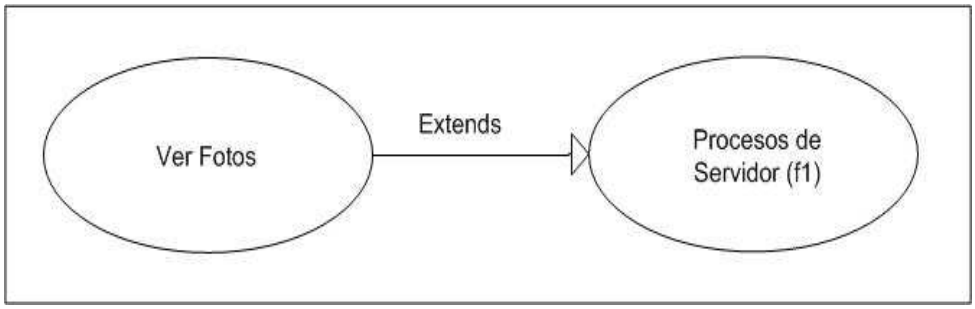
Elaborador por: Araceli Dávila M.

Gráfico No. 16 Casos de Uso Nivel II – Lectura de Noticias



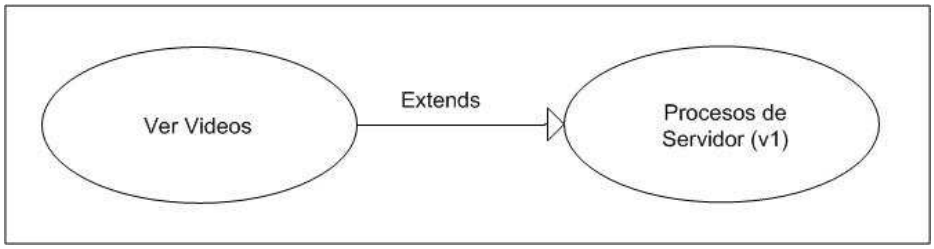
Elaborador por: Araceli Dávila M.

Gráfico No. 17 Casos de Uso Nivel II – Fotos



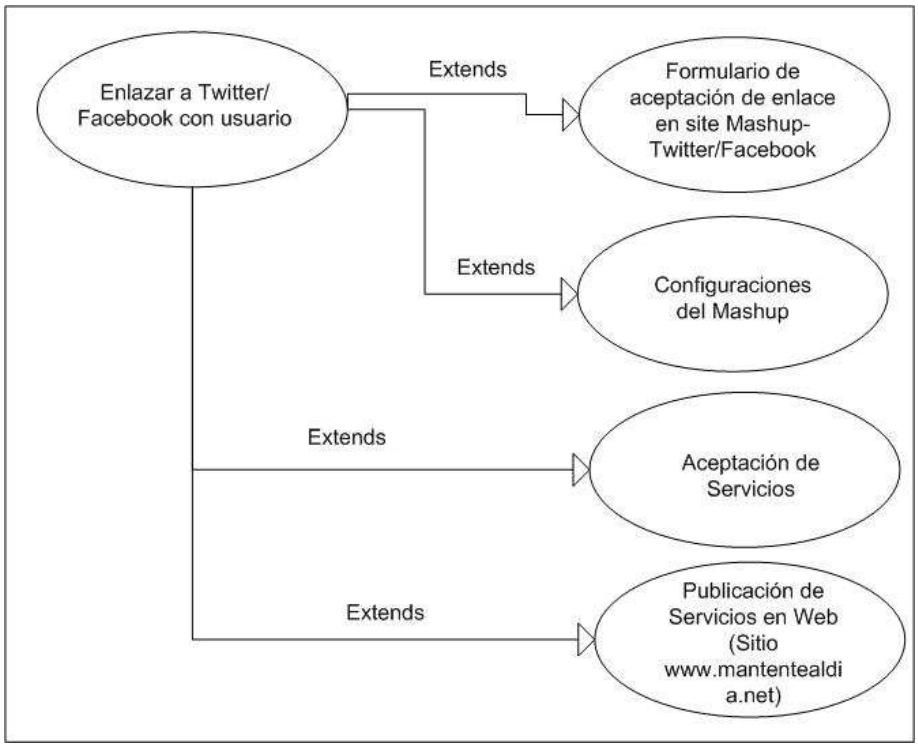
Elaborado por: Araceli Dávila M.

Gráfico No. 18 Casos de Uso Nivel II – Videos



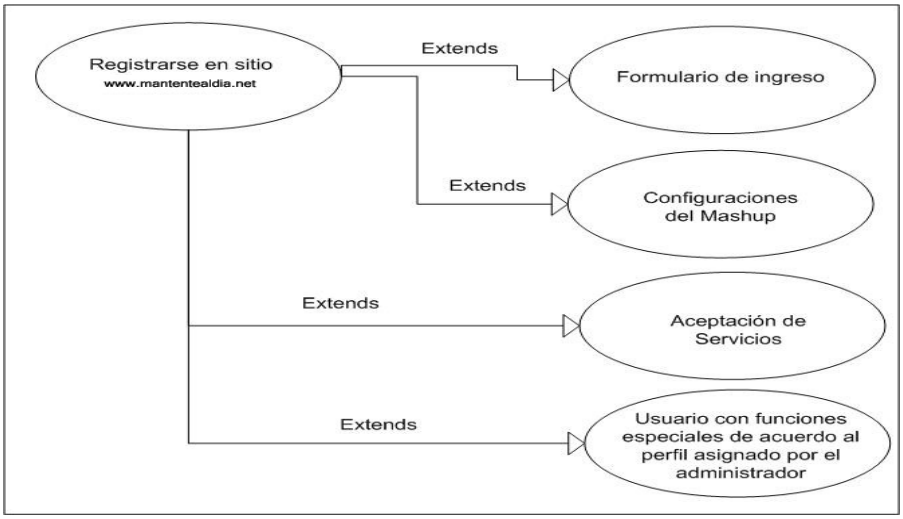
Elaborado por: Araceli Dávila M.

Gráfico No. 19 Casos de Uso Nivel II – Enlace con Módulo Twitter



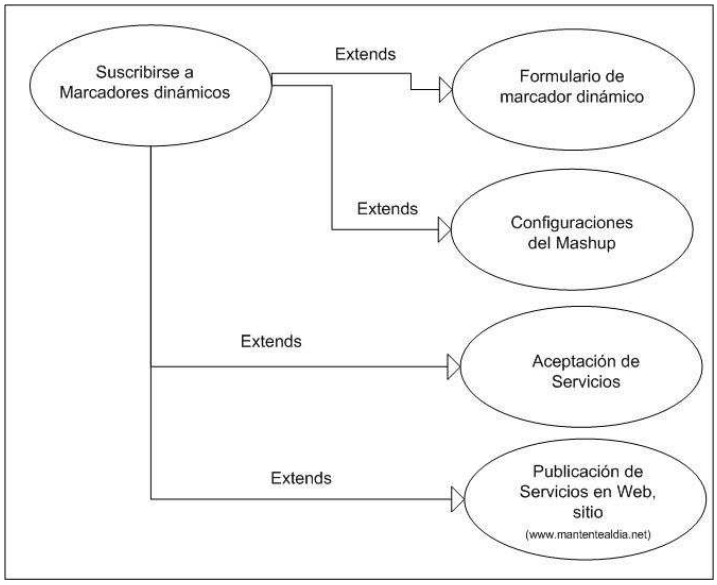
Elaborado por: Araceli Dávila M.

Gráfico No. 20 Casos de Uso Nivel II – Registro en Sitio



Elaborado por: Araceli Dávila M.

Gráfico No. 21 Casos de Uso Nivel II – Suscripción de Marcadores Dinámicos



Elaborado por: Araceli Dávila M.

Descripción de Casos de Uso

1

Caso de uso	Leer Noticias
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá navegar en el sitio web, es decir solo se mostrará el index del sitio

2

Caso de uso	Elaborar Encuestas
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permite mostrar los alcances de la aplicación al momento de realizar la encuesta de nuestro sitio.

3

Caso de uso	Ver Fotos
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá mostrar la iteracción con el servidor en lo concerniente a las fotos.

4

Caso de uso	Ver Videos
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá mostrar la iteracción con el servidor en lo concerniente a los videos.

5

Caso de uso	Leer Clasificados
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá mostrar lo concerniente al link de los avisos clasificados.

6

Caso de uso	Enviar mensajes al Administrador del Sitio
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá enviar mensajes a los administradores del sitio comenzando en el link correspondiente.

7

Caso de uso	Suscribirse a marcadores dinámicos
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá enlazar las Apis de marcadores dinámicos en el mashup de contenido

8

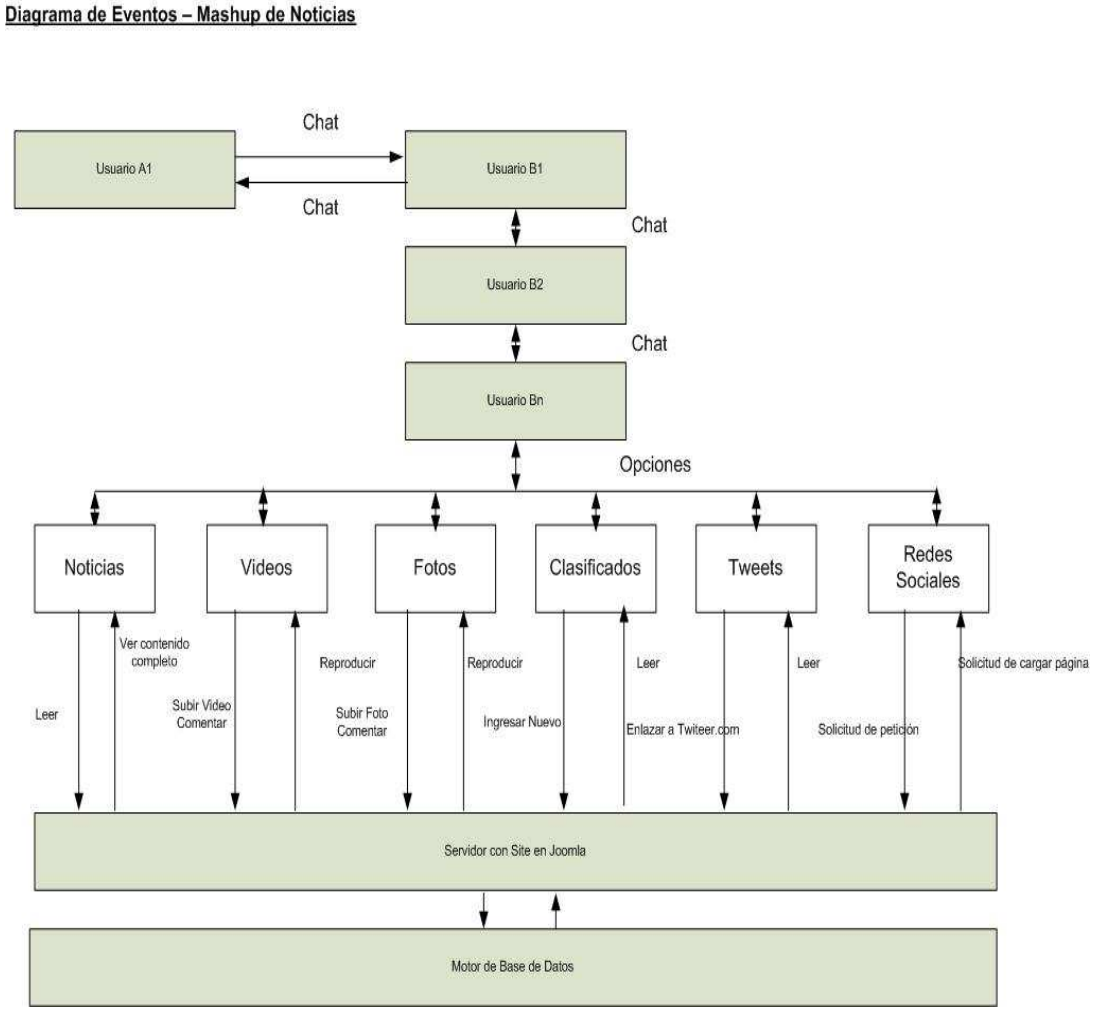
Caso de uso	Enlazar a Twitter/Facebook con un usuario
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá explicar el enlaze a las Apis de Twitter con nuestro Mashup de contenido.

9

Caso de uso	Registrarse en mantentealdia.net
Actores	Usuario, Bases de Datos
Tipo	Básico
Descripción	Este caso de uso me permitirá insertar, modificar la información de usuarios que deseen logearse a nuestra web y aprovechar las ventajas.

1.8 DIAGRAMA DE EVENTOS

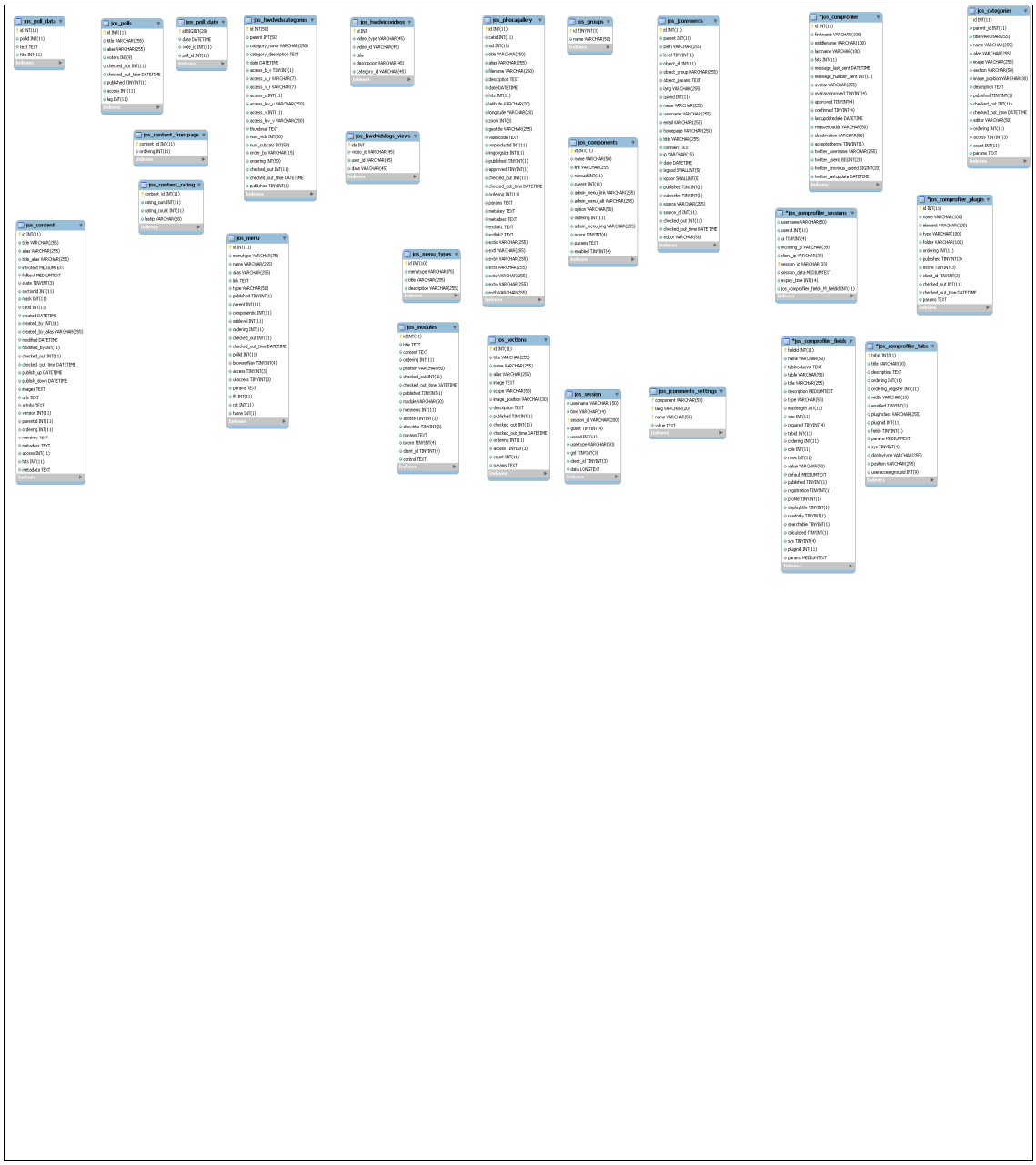
Gráfico No. 22 Diagrama de Eventos del Sitio



Elaborado por: Araceli Dávila

1.9 MODELO ENTIDAD RELACIÓN

Gráfico No. 23 MER Sitio Mashup



Elaborado por: Araceli Dávila M.

1.10 DICCIONARIO DE DATOS

Table jos_categories (Categorías del Sitio)

Este objeto guarda las distintas categorías que se pueden crear en una página web Joomla.

Joomla es un sistema de gestión de contenidos y entre sus principales virtudes está la de permitir editar el contenido de un sitio web de manera sencilla.

(1/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	parent_id	INT(11)	Yes	'0'	
	title	VARCHAR(255)	Yes	''	
	name	VARCHAR(255)	Yes	''	
	alias	VARCHAR(255)	Yes	''	
	image	VARCHAR(	Yes	''	

		255)			
	section	VARCHAR(50)	Yes	''	
	image_position	VARCHAR(30)	Yes	''	
	description	TEXT	Yes		
	published	TINYINT(1)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	editor	VARCHAR(50)	No	NULL	
	ordering	INT(11)	Yes	'0'	
	access	TINYINT(3)	Yes	'0'	
	count	INT(11)	Yes	'0'	
	params	TEXT	Yes		

Table jos_categorías = {id} + {parent_id} + { title} + {name} + alias + image + section + {image_position} + descripción + Published + {checked_out} +{ checked_out_time} + editor + {ordering} + {acess} + {count} + params.

id = {Clave primaria para el registro}, **Asignada por el servidor.

parent_id = {0}

title = {Titulo de la categoría del sitio web }

Name = {Nombre de la categoría del sitio web}

Alias = **Alias del titulo del sitio web

Image = **Imagen de la categoria del sitio web

Section = **Seccion de la categoria del sitio web

image_position = **La posición de la imagen de la categoria del sitio web (left)

Description = **La descripción de la imagen de la categoria del sitio web,

Published = **Lo publicado de la categoria por los usuarios del sitio web

checked_out = **Revisado fuera de la categoria del sitio web {0 }

checked_out_time = ** Revisado fuera de tiempo, valor por default {0}

Editor = **Editor de la categoria del sitio web

Ordering = **Orden de la categoria del sitio web, valor por default {1}

Access = **El acceso de la categoria del sitio web, valor por default {0 }

Count = **El contador de las categorias del sitio web, valor por default {0}

Params = **Los parámetros de la categoria del sitio web

carácter válido = [A-Z | a-z | ‘ |

Table jos_components (Componentes del Sitio)

Este objeto guarda los distintos componentes usados para el sitio web, Ejemplo calendario, Blogs, encuestas, Módulo Validador de Idioma, entre otros.

(2/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	Id	INT(11)	Yes		
	Name	VARCHAR(50)	Yes	''	
	Link	VARCHAR(255)	Yes	''	
	menuid	INT(11)	Yes	'0'	
	parent	INT(11)	Yes	'0'	
	admin_menu_link	VARCHAR(255)	Yes	''	
	admin_menu_alt	VARCHAR(255)	Yes	''	
	option	VARCHAR(50)	Yes	''	
	ordering	INT(11)	Yes	'0'	

	admin_menu_img	VARCHAR(255)	Yes	''	
	Iscore	TINYINT(4)	Yes	'0'	
	params	TEXT	Yes		
	enabled	TINYINT(4)	Yes	'1'	

Table jos_components = { id} + name + link + {menuid} +{parent} + admin_menu_link + admin_menu_alt + option + ordering + admin_menu_imag + iscore + params + enabled.

Id = **Clave primaria de la tabla de componentes del sitio web

Name = **Nombre de los componentes del sitio web

link = **Enlace al siguiente nivel de opciones

menuid = {0}

parent = {0}

admin_menu_link = **Enlace a las opciones que constan en el menu

admin_menu_alt = ** Gestion de banderas de los componentes del sitio Web

Option = **Opciones de los componentes del sitio web, banderas comunes

ordering = Niveles o escalas de los componentes del sitio web, por default {0}

admin_menu_imag = La admisión del menú de imagen

iscore = {0}

params = Los parámetros de los componentes del sitio web, pueden ser algunos.

Enabled = Habilitado a los componentes del sitio web (1)

carácter válido = [A-Z | a-z | ‘ |

Table *jos_comprofiler (Componentes instalados del Comunity Builder, para nuestro caso es el componente de Twitter)

Este objeto guarda la relación principal del componente de Twitter, donde está la fotografía del Avatar del usuario, y la ip de acceso.

El twitter es unade las redes sociales más famosas en la actualidad la cual sirve para estar en contacto con familiares, amigos, establecer negocios entre otras funciones.

Avatar es un campo que se usan en las tablas de la base de datos con el fin de realizar modifiaciones a las fotografías de las personas que visitan el sitio web.

La ip de acceso es una configuración que realiza el administrador del sitio con la finalidad de establecer y dar todos los permisos requeridos a las personas que ingresan al sitio web.

(3/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes	'0'	
	firstname	VARCHAR(100)	No	NULL	
	middlename	VARCHAR(100)	No	NULL	
	lastname	VARCHAR(100)	No	NULL	
	hits	INT(11)	Yes	'0'	
	message_last_sent	DATETIME	Yes	'0000-00-00 00:00:00'	
	message_number_sent	INT(11)	Yes	'0'	
	avatar	VARCHAR(255)	No	NULL	
	avatarapproved	TINYINT(4)	Yes	'1'	
	approved	TINYINT(4)	Yes	'1'	
	confirmed	TINYINT(4)	Yes	'1'	

	lastupdatedate	DATETIME	Yes	'0000-00-00 00:00:00'	
	registeripaddr	VARCHAR(50)	Yes	''	
	cbactivation	VARCHAR(50)	Yes	''	
	acceptedterms	TINYINT(1)	Yes	'0'	
	twitter_username	VARCHAR(255)	No	NULL	
	twitter_userid	BIGINT(20)	No	NULL	
	twitter_previous_userid	BIGINT(20)	No	NULL	
	twitter_lastupdate	DATETIME	Yes	'0000-00-00 00:00:00'	

jos_comprofiler=
id+{firstname}+{middlename}+{lastname}+hits+message_last_sent+
+message_number_sent+avatar+{avatarapproved}+{approved}+
+{confirmed}+lastupdatedate+registeripaddr+cbactivation+
+{acceptedterms}+twitter_username+twitter_userid+
+{twitter_previous_userid}+twitter_lastupdate

Id=**Valor Auto incremental dado por el servidor

firstname= {carácter válido}

middlename= {carácter válido}

lastname={carácter válido}

hits=**Número de Vistas

message_last_sent=**Fecha del ultimo mensaje de Twitter enviado

message_number_sent=**Cantidad de Twitter mensajes enviados

avatar=**Nombre del achivo de Fotografía del usuario de Twitter

avatarapproved={1}

approved={1}

confirmed={1}

lastupdatedate=**Fecha de última actualización

registeripaddr=**Registro de la Ip del visitante

cbactivation=**Activación del Community Builder

acceptedterms={0}

twitter_username=**Usuario de Twitter

twitter_userid=**Id de Twitter

twitter_previous_userid={NULL}

twitter_lastupdate=**Última fecha de actualización

carácter válido = [A-Z | a-z | ‘ |

Table *jos_comprofiler_fields (Campos de las tablas del sitio)

Esta tabla guarda información de los campos que participan en el sitio web, así como el tipo de uso y longitud de campos.

(4/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	fieldid	INT(11)	Yes		
	name	VARCHAR(50)	Yes	''	
	tablecolumns	TEXT	Yes		
	table	VARCHAR(50)	Yes	'#__comprofiler'	
	title	VARCHAR(255)	Yes	''	
	description	MEDIUMTEXT	Yes		
	type	VARCHAR(50)	Yes	''	
	maxlength	INT(11)	No	NULL	
	size	INT(11)	No	NULL	

	required	TINYINT(4)	No	'0'	
	tabid	INT(11)	No	NULL	
	ordering	INT(11)	No	NULL	
	cols	INT(11)	No	NULL	
	rows	INT(11)	No	NULL	
	value	VARCHAR(50)	No	NULL	
	default	MEDIUMTEXT	No	NULL	
	published	TINYINT(1)	Yes	'1'	
	registration	TINYINT(1)	Yes	'0'	
	profile	TINYINT(1)	Yes	'1'	
	displaytitle	TINYINT(1)	Yes	'1'	
	readonly	TINYINT(1)	Yes	'0'	
	searchable	TINYINT(1)	Yes	'0'	
	calculated	TINYINT(1)	Yes	'0'	
	sys	TINYINT(4)	Yes	'0'	
	pluginid	INT(11)	Yes	'0'	
	params	MEDIUMTEXT	No	NULL	

Jos_comprofiler_fields= { fielded} + name + table_columns + table + title + description + type + maxlenght + size + recorred + tableid + ordering + cols + rows + value + default + Published + registration + profile + displyatitle + readonly + searchable + calculated + sys + pluginid + paramas.

fieldid = Clave primaria de la tabla

Name = ** Nombre del campo

table_columns = **Información de las columnas

table = **Nombre del grupo de tabla perteneciente

Title = *Titulo de la tabla a la que se hace referencia

Description = **Descripción a la tabla a la que se hace referencia, para efectos del componente

Type = **Tipo

Maxlenght = **Longitud Máxima

size = **Tamaño, default Null

Requerid= **Requerido

Tableid = **Identificación de la tabla

Ordering = **Orden

Cols = **Columnas Null

Rows = **Filas Null

value = **Valores Null

default = **Por defecto {0}

Published = **Publicaciones (1)

registration = **Registro

Profile = **Perfil

Displaytitle = **Muestra el titulo, por default (1)

Readonly = **Solo para lectura {0}

Searchable = **Búsqueda disponible {0}

calculated = **Campo Calculado {0}

sys = **Campos de uso del Servidor

Pluginid = **Campos de uso del Servidor

Paramas = **Parametros, default Null

carácter válido = [A-Z | a-z | ‘ |

Table *jos_comprofiler_plugin (Componentes adicionales que funcionan con el Comunity Builder)

En este objeto se almacenan los componentes (pluggins) adicionales, para el funcionamiento exclusivo de adicionales para los componentes en mención. Ejemplo: Twiter Tab, CB Twitter.

Llamese Plugin, a un tipo de archivo en especial que se puede anexar a otro para aumentar la funcionalidad de un objeto inesperado.

EL CB Twitter, es el componente instalado actualmente para hacer el Mashup en la parte correspondiente del Twitter, debe su nombre a Comunity Builder Twitter.

El componente Twitter Tab, muestra los tweets en una determinada sección de la página, asi denominada Tab.

(5/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	name	VARCHAR(100)	Yes	"	
	element	VARCHAR(100)	Yes	"	
	type	VARCHAR(100)	No	"	
	folder	VARCHAR(100)	No	"	

	ordering	INT(11)	Yes	'0'	
	published	TINYINT(3)	Yes	'0'	
	iscore	TINYINT(3)	Yes	'0'	
	client_id	TINYINT(3)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	
	checked_out _time	DATETIME	Yes	'0000-00-00 00:00:00'	
	params	TEXT	Yes		

jos_comprofiler_plugin = { id} + {name} + element + type + folder + ordering + Published + {iscore} + {client_id} + {checked_out }+{ checked_out_time} +{ params}

Id = {Identificación del pk para la tabla}

Name = {Nombre}

element = **Elemento

Type = **Tipo (usuario)

Folder = **Carpetas

Ordering = **Orden de presentación del plugin (1)

Published = **Campo publicado(1)

iscore = {1}

Client _id = {0}

checked_out = **Chequeado fuera del proceso {0}

checked_out_time =** Chequeado fuera del proceso {0}

Params = **Parametros , pueden ser algunos

carácter válido = [A-Z | a-z | ‘ |

Table *jos_comprofiler_sessions (Manejador de sesiones en los components de Community Builder)

Esta tabla guarda la información relativa a la sesión de los usuarios, para su respectivo control.

(6/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
	username	VARCHAR(50)	Yes	''	
	userid	INT(11)	Yes	'0'	
	ui	TINYINT(4)	Yes	'0'	

	incoming_ip	VARCHAR(39)	Yes	''	
	client_ip	VARCHAR(39)	Yes	''	
PK	session_id	VARCHAR(33)	Yes	''	
	session_data	MEDIUMTEXT	Yes		
	expiry_time	INT(14)	Yes	'0'	

jos_comprofiler_sessions = {username} + userid + {ui} + incoming_ip + client_ip + session_id + session_data + expiry_time.

Username = **Nombre del usuario

userid = **Identificación del usuario {0}

Ui ={1}

incoming_ip = **Dirección Ip públicas de los usuarios que interactúan con el módulo

client_ip = **Dirección Ip del usuario que se conecta al sitio web

session_id = **identificación de sesiones

session_data = **Sesión otorgada por el servidor

expiry_time = **tiempo de expiración

carácter válido = [A-Z | a-z | ‘ |

Table *jos_comprofiler_tabs (Publicaciones de secciones de los componentes).

Esta tabla almacena los permisos a las publicaciones correspondientes por sección, para ser publicados o no. Ejemplo: My Tweets, que son los mensajes cortos enviados desde Twitter, o My Latest Tweets, que son el registro y publicación de los mensajes cortos enviados con anterioridad. Almacenándose desde el ultimo a al primero.

(7/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	tabid	INT(11)	Yes		
	title	VARCHAR(50)	Yes	''	
	description	TEXT	No	NULL	
	ordering	INT(11)	Yes	'0'	
	ordering_register	INT(11)	Yes	'10'	
	width	VARCHAR(10)	Yes	'.5'	
	enabled	TINYINT(1)	Yes	'1'	
	pluginclass	VARCHAR(255)	No	NULL	

	pluginid	INT(11)	No	NULL	
	fields	TINYINT(1)	Yes	'1'	
	params	MEDIUMTEXT	No	NULL	
	sys	TINYINT(4)	Yes	'0'	
	displaytype	VARCHAR(255)	Yes	''	
	position	VARCHAR(255)	Yes	''	
	useraccessgroupid	INT(9)	Yes	'-2'	

jos_comprofiler_tabs = { tabid }+ title + description + ordering + ordering_register + {width} + {enabled} +{ pluginclass} + pluginid + fields + params + {sys} + displaytype + position + useraccessgroupid.

tabid = **Clave primaria para identificar cada transacción

title = **titulo

Descripción =** Descripción

Ordering =** Orden de presentación del Tab

ordering_register =** Registro de ordenes

Width = {Ancho}, default (1)

Enabled = **Habilitado, default {1}

Pluginclass = {Proceso de Servidor}

pluginid = **Identificación del plugin

Fields = **Campos (1)

Params = **Parámetros Null

Sys ={Proceso de Servidor}

Displaytype = **Mostrando tipo

Position = **Posición

Useraccessgroupid = **Acceso al grupo de identificación de usuarios

carácter válido = [A-Z | a-z | ‘ |

Table jos_content.

Tabla de Contenidos, con respecto a todos los datos sin importar categorización.

(8/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		

	title	VARCHAR(255)	Yes	''	
	alias	VARCHAR(255)	Yes	''	
	title_alias	VARCHAR(255)	Yes	''	
	introtext	MEDIUMTEXT	Yes		
	fulltext	MEDIUMTEXT	Yes		
	state	TINYINT(3)	Yes	'0'	
	sectionid	INT(11)	Yes	'0'	
	mask	INT(11)	Yes	'0'	
	catid	INT(11)	Yes	'0'	
	created	DATETIME	Yes	'0000-00-00 00:00:00'	
	created_by	INT(11)	Yes	'0'	
	created_by_alias	VARCHAR(255)	Yes	''	
	modified	DATETIME	Yes	'0000-00-00 00:00:00'	
	modified_by	INT(11)	Yes	'0'	

	checked_out	INT(11)	Yes	'0'	
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	publish_up	DATETIME	Yes	'0000-00-00 00:00:00'	
	publish_down	DATETIME	Yes	'0000-00-00 00:00:00'	
	images	TEXT	Yes		
	urls	TEXT	Yes		
	attribs	TEXT	Yes		
	version	INT(11)	Yes	'1'	
	parentid	INT(11)	Yes	'0'	
	ordering	INT(11)	Yes	'0'	
	metakey	TEXT	Yes		
	metadesc	TEXT	Yes		
	access	INT(11)	Yes	'0'	
	hits	INT(11)	Yes	'0'	
	metadata	TEXT	Yes		

jos_content = { id} + title + alias + title_alias + introtext + fulltext + state + section_id +{ maks} + catid + created + created_by + created_by_alias + modified + modified_by + {checked_out} + checked_out_time + {publish_out} +{ publish_down} + images + urls + attribs + version + parentid + ordering + metakey + metadesc + {access} + hits + metadata.

Id = **Clave primaria de tabla

Title = **titulo del contenido del sitio web

Aias = **Alias del titulo de contenido del sitio web

title_alias =** Titulo de Alias del contenido del sitio web

Introtext = **Texto introductorio

Fulltext = **Texto normal

State = **Estado del contenido

section_id = **Identificación de secciones (1)

Maks = {0}

Catid = **Id de categoría

create = **Fecha de creación

created_by =** Creado por

created_by_alias = ** alias del usuario

Modified =** Modificado

modified_by = **Modificado por

checked_out = {0}

checked_out_time = **Publicado en la fecha

publish_out ={0}

publish_down ={0}

images = **Imágen adjunta

urls = **Urls

attribs = **Atributos

Version = **Versión

parentid = {0}

Ordering =** Ordenando {0}

metakey ={Para usos del servidor}

metadesc ={Para usos del servidor}

Access = **Aceso {0}

Hits = **Hits

metadata ={Para usos del servidor}

Table jos_content_frontpage (Contenido de la Cabezera de la Página, es decir la parte alta de la página.

Se usa en caso de tener algún Banner en la página. Llamese Banner a un anuncio rectangular ubicado muchas veces en la parte superior de la página.

(9/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	content_id	INT(11)	Yes	'0'	
	ordering	INT(11)	Yes	'0'	

Table jos_content_frontpage = { content_id }+ { ordering }

Content_id = ** { Primary Key para identificar el registro }

Ordering= ** { Orden que asigna el servidor a un registro }

Table jos_content_rating (Tabla del contenido del rating del contenido)

Esta tabla guarda la información de los artículos mas visitados, y la información relativa a ello.

(10/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	content_id	INT(11)	Yes	'0'	
	rating_sum	INT(11)	Yes	'0'	

	rating_count	INT(11)	Yes	'0'	
	lastip	VARCHAR(50)	Yes	"	

jos_content_rating = { content_id } + rating_sum + rating_count + { lastip}

Content_id = { Identificación de contenido }

Rating_sum = Clasificando la suma

Rating_count = Clasificando el conteo

Lastip = { Ultima dirección ip }

Table jos_groups (Tabla de Grupos del Sitio, entiéndase grupo por agrupación de elementos con o sin afinidad dentro un lugar).

(11/25)

Columns

Key	**Column Name**	**Datatype**	**Not Null**	**Default**	**Comment**
PK	id	TINYINT(3)	Yes	'0'	
	name	VARCHAR(50)	Yes	"	

jos_groups = { id } + { name}

Id = { Identificación de los grupos del sitio web } {0}

Name = { Nombre de los grupos del sitio web }

Table jos_hwdvidscategories (Categorías de Videos)

Esta tabla almacena información acerca de las subdivisiones de la página web, es decir categorías Al sitio se puede subir videos con extensiones .mpg, .3gp, .wmv y mp4, No existe Menu de Videos, solo una sección dedicada a subir y comentar los mismos.

(12/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(50)	Yes		
	parent	INT(50)	Yes	'0'	
	category_name	VARCHAR (250)	No	NULL	
	category_description	TEXT	No	NULL	

	date	DATETIME	Yes	'0000-00-00 00:00:00'	
	access_b_v	TINYINT(1)	Yes	'0'	
	access_u_r	VARCHAR(7)	Yes	'RECURSE'	
	access_v_r	VARCHAR(7)	Yes	'RECURSE'	
	access_u	INT(11)	Yes	'-2'	
	access_lev_u	VARCHAR(250)	Yes	'0,1'	
	access_v	INT(11)	Yes	'-2'	
	access_lev_v	VARCHAR(250)	Yes	'0,1'	
	thumbnail	TEXT	Yes		
	num_vids	INT(50)	Yes	'0'	
	num_subcats	INT(50)	Yes	'0'	
	order_by	VARCHAR(15)	Yes	'0'	
	ordering	INT(50)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	

	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	published	TINYINT(1)	Yes	'0'	

jos_hwdvidscategories = { id } + { parent } + category_name + category_description + date + { access_b_v } + { access_u_r } + { access_v_r } + { access_u } + { access_lev_u } + { access_v } + { {access_lev_v } +{ thumbnail } + num_vids + { num_subcats } + { order_by } + ordering + { checked_out } + { checked_out_time } + published.

Id = Identificación de la categoría de videos del sitio web

Parent = **En el caso de guardar relación de dos videos o mas.

category_name = Nombre de categorías

category_description = Descripción de categorías

Date = Fechas

access_b_v = { Iteración del servidor }

access_u_r = { Iteración del servidor }

access_v_r = { Iteración del servidor }

access_u = { Iteración del servidor }

access_lev_u = { Iteración del servidor }

access_v ={ Iteración del servidor }

access_lev_v ={ Iteración del servidor }

thumbnail = { Iteración del servidor }

num_vids = Numero de videos

num_subcats = {0}

order_by = Ordenar por {0}

Ordering = ordenes

checked_out = Revisa el video después de haber sido sido subido por el usuario {0}

checked_out_time = {0}

published = Publicaciones (1)

Table jos_jcomments (Tabla de comentarios del sitio)

Esta tabla esta creada con el único fin de que las personas que ingresen al sitio web y deseen emitir algún comentario se puedan guardar, esto con la aprobación del user Administrador.

(13/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	parent	INT(11)	Yes	'0'	
	path	VARCHAR(255)	Yes	''	
	level	TINYINT(1)	Yes	'0'	
	object_id	INT(11)	Yes	'0'	
	object_group	VARCHAR(255)	Yes	''	
	object_params	TEXT	Yes		
	lang	VARCHAR(255)	Yes	''	
	userid	INT(11)	Yes	'0'	
	name	VARCHAR(255)	Yes	''	
	username	VARCHAR(255)	Yes	''	

	email	VARCHAR(255)	Yes	''	
	homepage	VARCHAR(255)	Yes	''	
	title	VARCHAR(255)	Yes	''	
	comment	TEXT	Yes		
	ip	VARCHAR(15)	Yes	''	
	date	DATETIME	Yes	'0000-00-00 00:00:00'	
	isgood	SMALLINT(5)	Yes	'0'	
	ispoor	SMALLINT(5)	Yes	'0'	
	published	TINYINT(1)	Yes	'0'	
	subscribe	TINYINT(1)	Yes	'0'	
	source	VARCHAR(255)	Yes	''	
	source_id	INT(11)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	
	checked_out_	DATETIME	Yes	'0000-00-00	

	time			00:00:00'	
	editor	VARCHAR(50)	Yes	"	

jos_jcomments = { id } + { parent }+ { path }+ { level } + object_id + object_group + object_params + lang + { userid } + { name } +{ username } + email + homepage + { title} + comment + ip + date + { Isgood} + { ispoor }+ Published + { subscribe } + source + { source_id } + { checked_out } + { checked_out_time } + editor

Id = Identificación de los comentarios del sitio web

Parent = {0}

Path = { Ruta de los comentarios del sitio web } {0}

Level = { Nivel de los comentarios del sitio web } {0}

object_id = Identificación del objeto de los comentarios del sitio web (1)

object_group = Grupo de objeto de los comentarios del sitio web

object_params = Objeto de parametros de los comentarios del sitio web

Lang = ** Lenguaje manejado en el comentario

Userid = { Identificación del usuario de los comentarios del sitio web }

Name = { Nombre de los comentarios del sitio web }

Username = { Nombre de usuario de los comentarios del sitio web }

Email = Correo en el caso de que el usuario decida ingresarlo al sitio web

Homepage = Pagina de Inicio del sitio web

Title = { Titulo del sitio web }

Comment = Comentario del sitio web

Ip = Direccion Ip del sitio web

Date = Fecha del sitio web

isgood = { Es bueno } {0}

Ispoor = { Es pobre } {0}

Published = Publicaciones (1)

Subscribe = Suscripciones {0}

Source = fuente

source_id = { Identificación de la fuente } {0}

checked_out = {Interacción con el servidor]{0}

checked_out_time = {Interacción con el servidor } {0}

Editor = Editor

Table jos_jcomments_settings (Tabla de configuración del comentario)

Es decir información relevante al comentario de un referido video, imagen, o noticia. Llamese referido al comentario de un usuario.

(14/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	Component	VARCHAR(50)	Yes	''	
PK	Lang	VARCHAR(20)	Yes	''	
PK	Name	VARCHAR(50)	Yes	''	
	Value	TEXT	Yes		

jos_jcomments_settings = { component } + { lang } + { name} + value

Component = { Componentes del sitio web y clave primaria }

Lang = { Lenguaje Wapa y clave primaria }

Name = { Nombres del sitio web y clave primaria }

Value = Valores del sitio web

Table jos_menu (Tabla de Menus)

Información relevante a los menus del sitio web.

(15/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	Id	INT(11)	Yes		
	Menutype	VARCHAR(75)	No	NULL	
	Name	VARCHAR(255)	No	NULL	
	Alias	VARCHAR(255)	Yes	''	
	Link	TEXT	No	NULL	
	Type	VARCHAR(50)	Yes	''	
	Published	TINYINT(1)	Yes	'0'	
	Parent	INT(11)	Yes	'0'	
	Componentid	INT(11)	Yes	'0'	
	Sublevel	INT(11)	No	'0'	
	Ordering	INT(11)	No	'0'	
	checked_ou	INT(11)	Yes	'0'	

	t				
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	Pollid	INT(11)	Yes	'0'	
	browserNav	TINYINT(4)	No	'0'	
	Access	TINYINT(3)	Yes	'0'	
	Utaccess	TINYINT(3)	Yes	'0'	
	Params	TEXT	Yes		
	Lft	INT(11)	Yes	'0'	
	Rgt	INT(11)	Yes	'0'	

jos_menu = { id } + { menutype} + {name }+ { alias } + link + type + Publisher + { parent } + componentid + { sublevel } + ordering +{ checked_out } + { checked_out_time } + { pollid } + { browserNav }+ { access } + { utaccess } + params + { lft } +{ rgt }

Id = { Identificación del menu del sitio web }

Menutype = { Menu principal del sitio web }

Name = { Nombre del sitio web }

Alias = { Alias del nombre del sitio web }

Link = **La Url a donde te llevará el sitio en caso de hacer click.

Type = Tipo

Published = Publicaciones (1)

Parent ={0}

Componentid = { Identificaciones de componentes }

Sublevel = {0}

Ordering = Órdenes

checked_out = {0}

checked_out_time = {0}

Pollid ={0}

browserNav ={0}

access = Acceso {0}

Utaccess = {Valores asignados por el servidor }

Params = Parámetros

Lft = Izquierda {0}

Rgt = Derecha {0}

Table jos_menu_types (Tabla de Tipos de Menús)

En esta tabla se encontraran los diferentes tipos de menú que se encontraran en el sitio web.

(16/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	Id	INT(10)	Yes		
	Menutype	VARCHAR(75)	Yes	"	
	Title	VARCHAR(255)	Yes	"	
	Description	VARCHAR(255)	Yes	"	

jos_menu_types ={ id }+ { menutype} + { title} +{ description }

Id = { Identificación del tipo de menú del sitio web }

Menutype = { Tipo de menú del sitio web }

Title = { Titulo del sitio web }

Description = { Descripción del sitio web }

Table jos_modules (Tabla de Módulos)

Son aquellos que se encuentran instalados para el efecto que queramos dar a nuestro sitio. En el caso de no tenerlo, se lo debe instalar previamente.

(17/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	title	TEXT	Yes		
	content	TEXT	Yes		
	ordering	INT(11)	Yes	'0'	
	position	VARCHAR(50)	No	NULL	
	checked_out	INT(11)	Yes	'0'	
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	published	TINYINT(1)	Yes	'0'	
	module	VARCHAR(50)	No	NULL	
	numnews	INT(11)	Yes	'0'	
	access	TINYINT(3)	Yes	'0'	
	showtitle	TINYINT(3)	Yes	'1'	
	params	TEXT	Yes		
	iscore	TINYINT(4)	Yes	'0'	

	client_id	TINYINT(4)	Yes	'0'	
	control	TEXT	Yes		

jos_modules = { id } + { title } + content + ordering + position + { checked_out} +{ checked_out_time} + Published + module + { numnews } + { access } + showtitle + params + iscore + { client_id } + control }

Id = { Identificación de los módulos del sitio web }

Title = { Titulo de los módulos los sitio web }

Content = Contenido de los módulos del sitio web

Ordering = Órdenes de los módulos del sitio web

Position = Posiciones de los módulos del sitio web

checked_out = {0}

checked_out_time = { 0}

Published = Publicaciones de los módulos del sitio web (1)

Module = Módulos del sitio web

Numnews = { Numero de noticias del sitio web } {0}

Access = { Acceso de los módulos del sitio web } {0}

showtitle = Mostrando el titulo (1)

Params = Parámetros

Iscore = (1)

client_id = { Identificación del cliente } {0}

Control = Control

Table jos_phocagallery (Tabla de Información de Fotografías)

Aquí podemos encontrar todo tipo de información referente a la galeria de fotografías es decir decir, título, alias, descripción. Fecha, tamaño de la imagen, etc.

(18/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	catid	INT(11)	Yes	'0'	
	sid	INT(11)	Yes	'0'	
	title	VARCHAR(250)	Yes	''	
	alias	VARCHAR(255)	Yes	''	
	filename	VARCHAR(250)	Yes	''	

	description	TEXT	No	NULL	
	date	DATETIME	Yes	'0000-00-00 00:00:00'	
	hits	INT(11)	Yes	'0'	
	latitude	VARCHAR(20)	Yes	''	
	longitude	VARCHAR(20)	Yes	''	
	zoom	INT(3)	Yes	'0'	
	geotitle	VARCHAR(255)	Yes	''	
	videocode	TEXT	No	NULL	
	vmproductid	INT(11)	Yes	'0'	
	imgorigsize	INT(11)	Yes	'0'	
	published	TINYINT(1)	Yes	'0'	
	approved	TINYINT(1)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	ordering	INT(11)	Yes	'0'	
	params	TEXT	No	NULL	

	metakey	TEXT	No	NULL	
	metadesc	TEXT	No	NULL	
	extlink1	TEXT	No	NULL	
	extlink2	TEXT	No	NULL	
	extid	VARCHAR(255)	Yes	"	
	extl	VARCHAR(255)	Yes	"	
	extm	VARCHAR(255)	Yes	"	
	exts	VARCHAR(255)	Yes	"	
	exto	VARCHAR(255)	Yes	"	
	extw	VARCHAR(255)	Yes	"	
	exth	VARCHAR(255)	Yes	"	

jos_phocagallery = { id } + { catid } + { sid } + { title } +{ alias } + fílenme + description + date + hits + latitude + longitude + zoom + geotitle + videocode + { vmproductid } + imgorigsize + published + approved + { checked_out } + { checked_out_time } + ordering + {params }+ { metakey }+ { metadesc } + {

extlink1 } + { exlink2 } + { extid } + { extl } + { extm } + exts + { exto } + { extw } +{ exth }

Id = { Identificación de la galería de fotos del sitio web }

Catid = { Asignado por el servidor } (1)

Sid = {0}

Title = { Titulo de la galería de fotos del sitio web }

Alias = { Alias de la galería de fotos del sitio web }

Filename = Nombre del archivo de la galería de fotos del sitio web

description = Descripción de la galería de fotos del sitio web

Date = Fecha de la galería de fotos del sitio web

Hits = {0}

Latitude = Latitud de la galería de fotos del sitio web

Longitude = Longitud de la galería de fotos del sitio web

Zoom = Zoom de la galería de fotos del sitio web

Geotitle = Información geográfica de la foto del sitio web

Videocode = Codigo del video Null

vmproductid = {0}

Imgorigsize = Tamaño original de la imagen

Published = Publicación de la galería de fotos del sitio web (1)

Approved = (1)

checked_out = {0}

checked_out_time = {0}

Ordering = Órdenes de la galería de fotos del sitio web (1)

Params = {Interacción por el servidor } Parámetros Null

Metakey = {Interacción por el servidor } Null

metadesc = {Interacción por el servidor } Null

extlink1 = {Interacción por el servidor } Null

Extlink2 = {Interacción por el servidor } Null

Extid = {Interacción por el servidor }

Extl = {Interacción por el servidor }

Extm = {Interacción por el servidor }

Exts = Extensiones de la galería de fotos del sitio web

Exto = {Interacción por el servidor }

Extw = {Interacción por el servidor }

Exth ={Interacción por el servidor }

Table jos_poll_data (Tabla de Preguntas de Encuestas)

Esta tabla guarda información solo de las preguntas, en caso de publicar una encuesta.

(19/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	pollid	INT(11)	Yes	'0'	
	text	TEXT	Yes		
	hits	INT(11)	Yes	'0'	

jos_poll_data = { id } + { pollid } + text + hits

Id = { Identificación de los datos de las encuestas y clave primaria del sitio web }

Pollid = { Id de la encuesta del sitio web }

Text = Texto de los datos del las encuestas del sitio web

Hits = Número de veces en que se ha realizado la encuesta

Table jos_poll_date (Tabla de respuestas de encuestas)

En esta tabla se almacenará el banco de respuestas, sin importar el orden de las preguntas.

(20/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	BIGINT(20)	Yes		
	date	DATETIME	Yes	'0000-00-00 00:00:00'	
	vote_id	INT(11)	Yes	'0'	
	poll_id	INT(11)	Yes	'0'	

jos_poll_date = { id }+ date + vote_id + {poll_id }

Id = { Identificación de las encuestas por fecha del sitio web }

Date = Fechas de las encuestas del sitio web

Vote_id = Identificación de Votos de las encuestas por fecha del sitio web

Poll_id = { identificación de las encuesta del sitio web }

Table jos_polls (Tabla de Información de la encuesta)

Esta tabla almacena, la información estadística de la encuesta, es decir los votos, la fecha del alta=fecha en que si el usuario Administrador da el ok para subirla al sitio en producción a la publicación.

(21/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT(11)	Yes		
	title	VARCHAR(255)	Yes	''	
	alias	VARCHAR(255)	Yes	''	
	voters	INT(9)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	published	TINYINT(1)	Yes	'0'	
	access	INT(11)	Yes	'0'	
	lag	INT(11)	Yes	'0'	

jos_polls = { id } + { title } + { alias }+ voters + { checked_out } + { checked_out_time } + Published + { access } + lag.

Id = { Identificación de las encuestas del sitio web}

Title = { Título de las encuestas del sitito web }

Alias = { Alias del título de la pregunta de la encuesta del sitio web }

Voters = Votantes de las encuestas del sitio web

checked_out = {0}

checked_out_time = {0}

Published = Publicaciones de las encuestas (1)

Access = Acceso de las encuestas del sitio web [0}

Lag = Es asignado por el servidor

Table jos_sections (Tabla de secciones)

En Joomla se manejan también Secciones a las subdivisiones de contenidos, esta tabla almacena los nombres de las mismas para el respectivo control independiente.

(22/25)

Columns

Key	**Column Name**	**Datatype**	**Not Null**	**Default**	**Comment**
PK	Id	INT(11)	Yes		
	Title	VARCHAR(255)	Yes	"	
	Name	VARCHAR(2	Yes	"	

		55)			
	Alias	VARCHAR(255)	Yes	"	
	Image	TEXT	Yes		
	Scope	VARCHAR(50)	Yes	"	
	image_position	VARCHAR(30)	Yes	"	
	Description	TEXT	Yes		
	Published	TINYINT(1)	Yes	'0'	
	checked_out	INT(11)	Yes	'0'	
	checked_out_time	DATETIME	Yes	'0000-00-00 00:00:00'	
	Ordering	INT(11)	Yes	'0'	
	Access	TINYINT(3)	Yes	'0'	
	Count	INT(11)	Yes	'0'	
	Params	TEXT	Yes		

jos_sections = { id } + { title } +{ name } + { alias } + image + scope + image_position + description + Published + { checked_out } + { checked_out_time } + ordering + { access } + count + params.

Id = { Identificación de las secciones del sitio web }

Title = { Título de las secciones del sitio web }

Name = { Nombre de las secciones del sitio web }

Alias = { Alias del campo del titulo de las secciones }

Image = Imagen de las secciones del sitio web

Scope = Contenido de las secciones del sitio web

image_position = Posición de la imagen

description = Descripción de las secciones del sitio web

Published = Publicaciones de las secciones del sitio web

checked_out = {0}

checked_out_time = {0}

Ordering = Órdenes de las secciones del sitio web

Access = Acceso de las secciones del sitio web {0}

Count = Contadores de las secciones del sitio web

Params = Parámetros de las secciones del sitio web

Table jos_session (Tabla de Sesiones)

Este objeto almacena toda la información necesaria con respecto a la sesión de un usuario “x”.

Se procede a dar la denominación de usuario “x” a cualquier tipo de persona que ingrese al sitio Web.

(23/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
	Username	VARCHAR(150)	No	''	
	Time	VARCHAR(14)	No	''	
PK	session_id	VARCHAR(200)	Yes	'0'	
	Guest	TINYINT(4)	No	'1'	
	Userid	INT(11)	No	'0'	
	Usertype	VARCHAR(50)	No	''	
	Gid	TINYINT(3)	Yes	'0'	
	client_id	TINYINT(3)	Yes	'0'	
	Data	LONGTEXT	No	NULL	

jos_session = { username } + time +{ session_id } + guest + { userid } + { usertype} + gid +{ client_id } + { data }

Username = { Nombre de Usuario }

Time = Tiempo

Session_id = { Identificación de sesiones y clave primaria }

Guest = (1)

Userid = { Identificación de usuario } {0}

Usertype = Tipo de usuario

Gid = {0}

Client_id = { Identificación del cliente } (1)

Data = { Datos de las sesiones del sitio web }

Table jos_hwdvidsvideos (Tabla de Manejador de Videos)

Esta tabla guarda información relevante a datos adicionales de los videos, es decir: título, descripción, tipo si es de youtube, duración, etc.

(24/25).

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	id	INT	Yes		
	video_type	VARCHAR(45)	No		
	video_id	VARCHAR(45)	No		
	title		No		
	descripcion	VARCHAR(45)	No		
	category_id	VARCHAR(45)	No		

jos_hwdvidsvideos ={ id } + { video_type } + { video_id } + { title } + { descripción } + { category_id }

Id = { Identificación de los videos del sitio web }

Video_type = { Tipo de video del sitio web }

Video_id = { Identificación del video del sitio web }

Title = { Titulo del stitio web }

Description = { Descripción del video del sitio web }

category_id = { Identificación de categoría del video del sitio web }

Table jos_hwdvidslogs_views (Tabla de información de Videos)

En esta tabla se guarda el detalle de la estadística del mismo.

(25/25)

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	ide	INT	Yes		
	video_id	VARCHAR(45)	No		
	user_id	VARCHAR(45)	No		
	date	VARCHAR(45)	No		

jos_hwdvidslogs_views = { ide } + { video_id } + { user_id } + { date }

Ide = { Identificación }

Video_id = { Identificación de video del sitio web }

User_id = { Identificación de usuario del sitio web }

Date = { Fecha del sitio web }

1.11 SCRIPT PARA LA CREACIÓN DE BASE DE DATOS

```
-- phpMyAdmin SQL Dump
-- version 3.3.9
-- http://www.phpmyadmin.net

-- Servidor: localhost
-- Tiempo de generacion: 20-02-2011 a las 12:31:33
-- Version del servidor: 5.1.52
-- Version de PHP: 5.2.4

-- Creación de Base de datos
-- Base de datos: `mashupdb`
-- create database mashupdb;

use mashupdb;

-- Estructura de tabla para la tabla `jos_categories`
-- Categorías del sitio


CREATE TABLE IF NOT EXISTS `jos_categories`
 (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `parent_id` int(11) NOT NULL DEFAULT '0',
  `title` varchar(255) NOT NULL DEFAULT '',
```

```
  `name` varchar(255) NOT NULL DEFAULT '',
  `alias` varchar(255) NOT NULL DEFAULT '',
  `image` varchar(255) NOT NULL DEFAULT '',
  `section` varchar(50) NOT NULL DEFAULT '',
  `image_position` varchar(30) NOT NULL DEFAULT '',
  `description` text NOT NULL,
  `published` tinyint(1) NOT NULL DEFAULT '0',
  `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `editor` varchar(50) DEFAULT NULL,
  `ordering` int(11) NOT NULL DEFAULT '0',
  `access` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `count` int(11) NOT NULL DEFAULT '0',
  `params` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `cat_idx` (`section`,`published`,`access`),
  KEY `idx_access` (`access`),
  KEY `idx_checkout` (`checked_out`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=50 ;

--
-- Estructura de tabla para la tabla `jos_components`
-- Componentes del Sitio


CREATE TABLE IF NOT EXISTS `jos_components`
 (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL DEFAULT '',
```

```
  `link` varchar(255) NOT NULL DEFAULT '',
  `menuid` int(11) unsigned NOT NULL DEFAULT '0',
  `parent` int(11) unsigned NOT NULL DEFAULT '0',
  `admin_menu_link` varchar(255) NOT NULL DEFAULT '',
  `admin_menu_alt` varchar(255) NOT NULL DEFAULT '',
  `option` varchar(50) NOT NULL DEFAULT '',
  `ordering` int(11) NOT NULL DEFAULT '0',
  `admin_menu_img` varchar(255) NOT NULL DEFAULT '',
  `iscore` tinyint(4) NOT NULL DEFAULT '0',
  `params` text NOT NULL,
  `enabled` tinyint(4) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  KEY `parent_option` (`parent`,`option`(32))
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=103 ;

--
-- Estructura de tabla para la tabla `jos_comprofiler`
--


CREATE TABLE IF NOT EXISTS `jos_comprofiler`
(
  `id` int(11) NOT NULL DEFAULT '0',
  `user_id` int(11) NOT NULL DEFAULT '0',
  `firstname` varchar(100) DEFAULT NULL,
  `middlename` varchar(100) DEFAULT NULL,
  `lastname` varchar(100) DEFAULT NULL,
  `hits` int(11) NOT NULL DEFAULT '0',
  `message_last_sent` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
```

```
  `message_number_sent` int(11) NOT NULL DEFAULT '0',
  `avatar` varchar(255) DEFAULT NULL,
  `avatarapproved` tinyint(4) NOT NULL DEFAULT '1',
  `approved` tinyint(4) NOT NULL DEFAULT '1',
  `confirmed` tinyint(4) NOT NULL DEFAULT '1',
  `lastupdatedate` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `registeripaddr` varchar(50) NOT NULL DEFAULT '',
  `cbactivation` varchar(50) NOT NULL DEFAULT '',
  `banned` tinyint(4) NOT NULL DEFAULT '0',
  `banneddate` datetime DEFAULT NULL,
  `unbanneddate` datetime DEFAULT NULL,
  `bannedby` int(11) DEFAULT NULL,
  `unbannedby` int(11) DEFAULT NULL,
  `bannedreason` mediumtext,
  `acceptedterms` tinyint(1) NOT NULL DEFAULT '0',
  `twitter_username` varchar(255) DEFAULT NULL,
  `twitter_userid` bigint(20) unsigned DEFAULT NULL,
  `twitter_previous_userid` bigint(20) unsigned DEFAULT NULL,
  `twitter_lastupdate` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `twUsername` varchar(255) DEFAULT NULL,
  `twittername` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `user_id` (`user_id`),
  KEY `apprconfbanid` (`approved`,`confirmed`,`banned`,`id`),
  KEY `avatappr_apr_conf_ban_avatar`
(`avatarapproved`,`approved`,`confirmed`,`banned`,`avatar`),
  KEY `lastupdatedate` (`lastupdatedate`),
  KEY `twitter_userid` (`twitter_userid`),
  KEY `twitter_previous_userid` (`twitter_previous_userid`)
)
```

```
ENGINE=MyISAM DEFAULT CHARSET=utf8;


-- Estructura de tabla para la tabla `jos_comprofiler_fields`
-- (Campos de las tablas del sitio)


CREATE TABLE IF NOT EXISTS `jos_comprofiler_fields`
 (
  `fieldid` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL DEFAULT '',
  `tablecolumns` text NOT NULL,
  `table` varchar(50) NOT NULL DEFAULT '#__comprofiler',
  `title` varchar(255) NOT NULL DEFAULT '',
  `description` mediumtext NOT NULL,
  `type` varchar(50) NOT NULL DEFAULT '',
  `maxlength` int(11) DEFAULT NULL,
  `size` int(11) DEFAULT NULL,
  `required` tinyint(4) DEFAULT '0',
  `tabid` int(11) DEFAULT NULL,
  `ordering` int(11) DEFAULT NULL,
  `cols` int(11) DEFAULT NULL,
  `rows` int(11) DEFAULT NULL,
  `value` varchar(50) DEFAULT NULL,
  `default` mediumtext,
  `published` tinyint(1) NOT NULL DEFAULT '1',
  `registration` tinyint(1) NOT NULL DEFAULT '0',
  `profile` tinyint(1) NOT NULL DEFAULT '1',
  `displaytitle` tinyint(1) NOT NULL DEFAULT '1',
  `readonly` tinyint(1) NOT NULL DEFAULT '0',
```

```
  `searchable` tinyint(1) NOT NULL DEFAULT '0',
  `calculated` tinyint(1) NOT NULL DEFAULT '0',
  `sys` tinyint(4) NOT NULL DEFAULT '0',
  `pluginid` int(11) NOT NULL DEFAULT '0',
  `params` mediumtext,
  PRIMARY KEY (`fieldid`),
  KEY `tabid_pub_prof_order` (`tabid`,`published`,`profile`,`ordering`),
  KEY `readonly_published_tabid` (`readonly`,`published`,`tabid`),
  KEY `registration_published_order` (`registration`,`published`,`ordering`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=60 ;
```

-- Estructura de tabla para la tabla `jos_comprofiler_plugin`

-- (Componentes adicionales que funcionan con el Comunity Builder)

```
CREATE TABLE IF NOT EXISTS `jos_comprofiler_plugin`
(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL DEFAULT '',
  `element` varchar(100) NOT NULL DEFAULT '',
  `type` varchar(100) DEFAULT '',
  `folder` varchar(100) DEFAULT '',
  `backend_menu` varchar(255) NOT NULL DEFAULT '',
  `access` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `ordering` int(11) NOT NULL DEFAULT '0',
  `published` tinyint(3) NOT NULL DEFAULT '0',
  `iscore` tinyint(3) NOT NULL DEFAULT '0',
  `client_id` tinyint(3) NOT NULL DEFAULT '0',
  `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
```

```
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `params` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_folder` (`published`,`client_id`,`access`,`folder`),
  KEY `type_pub_order` (`type`,`published`,`ordering`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=504 ;
```

-- Estructura de tabla para la tabla `jos_comprofiler_sessions`

-- (Manejador de sesiones en los components de Community Builder)

```
CREATE TABLE IF NOT EXISTS `jos_comprofiler_sessions`
(
  `username` varchar(50) NOT NULL DEFAULT '',
  `userid` int(11) unsigned NOT NULL DEFAULT '0',
  `ui` tinyint(4) NOT NULL DEFAULT '0',
  `incoming_ip` varchar(39) NOT NULL DEFAULT '',
  `client_ip` varchar(39) NOT NULL DEFAULT '',
  `session_id` varchar(33) NOT NULL DEFAULT '',
  `session_data` mediumtext NOT NULL,
  `expiry_time` int(14) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`session_id`),
  KEY `expiry_time` (`expiry_time`),
  KEY `userid` (`userid`)
)
ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
--
-- Estructura de tabla para la tabla `jos_comprofiler_tabs`
--

CREATE TABLE IF NOT EXISTS `jos_comprofiler_tabs`
 (
  `tabid` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(50) NOT NULL DEFAULT '',
  `description` text,
  `ordering` int(11) NOT NULL DEFAULT '0',
  `ordering_register` int(11) NOT NULL DEFAULT '10',
  `width` varchar(10) NOT NULL DEFAULT '.5',
  `enabled` tinyint(1) NOT NULL DEFAULT '1',
  `pluginclass` varchar(255) DEFAULT NULL,
  `pluginid` int(11) DEFAULT NULL,
  `fields` tinyint(1) NOT NULL DEFAULT '1',
  `params` mediumtext,
  `sys` tinyint(4) NOT NULL DEFAULT '0',
  `displaytype` varchar(255) NOT NULL DEFAULT '',
  `position` varchar(255) NOT NULL DEFAULT '',
  `useraccessgroupid` int(9) NOT NULL DEFAULT '-2',
  PRIMARY KEY (`tabid`),
  KEY `enabled_position_ordering` (`enabled`,`position`,`ordering`),
  KEY `orderreg_enabled_pos_order`
(`enabled`,`ordering_register`,`position`,`ordering`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=25 ;
```

-- Estructura de tabla para la tabla `jos_content`

-- (Tabla de Contenidos)

```
CREATE TABLE IF NOT EXISTS `jos_content`
 (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL DEFAULT '',
  `alias` varchar(255) NOT NULL DEFAULT '',
  `title_alias` varchar(255) NOT NULL DEFAULT '',
  `introtext` mediumtext NOT NULL,
  `fulltext` mediumtext NOT NULL,
  `state` tinyint(3) NOT NULL DEFAULT '0',
  `sectionid` int(11) unsigned NOT NULL DEFAULT '0',
  `mask` int(11) unsigned NOT NULL DEFAULT '0',
  `catid` int(11) unsigned NOT NULL DEFAULT '0',
  `created` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `created_by` int(11) unsigned NOT NULL DEFAULT '0',
  `created_by_alias` varchar(255) NOT NULL DEFAULT '',
  `modified` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `modified_by` int(11) unsigned NOT NULL DEFAULT '0',
  `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `publish_up` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `publish_down` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `images` text NOT NULL,
  `urls` text NOT NULL,
  `attribs` text NOT NULL,
  `version` int(11) unsigned NOT NULL DEFAULT '1',
  `parentid` int(11) unsigned NOT NULL DEFAULT '0',
```

```
  `ordering` int(11) NOT NULL DEFAULT '0',
  `metakey` text NOT NULL,
  `metadesc` text NOT NULL,
  `access` int(11) unsigned NOT NULL DEFAULT '0',
  `hits` int(11) unsigned NOT NULL DEFAULT '0',
  `metadata` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_section` (`sectionid`),
  KEY `idx_access` (`access`),
  KEY `idx_checkout` (`checked_out`),
  KEY `idx_state` (`state`),
  KEY `idx_catid` (`catid`),
  KEY `idx_createdby` (`created_by`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=134 ;
```

-- Estructura de tabla para la tabla `jos_content_frontpage`

-- (COntenido de la Cabezera de la Página)

```
CREATE TABLE IF NOT EXISTS `jos_content_frontpage`
 (
  `content_id` int(11) NOT NULL DEFAULT '0',
  `ordering` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`content_id`)
)
ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

--

```
-- Volcar la base de datos para la tabla `jos_content_frontpage`
--

INSERT INTO `jos_content_frontpage` (`content_id`, `ordering`) VALUES
(75, 1);


--
-- Estructura de tabla para la tabla `jos_content_rating`
-- (Tabla del contenido del rating del contenido)


CREATE TABLE IF NOT EXISTS `jos_content_rating`
(
  `content_id` int(11) NOT NULL DEFAULT '0',
  `rating_sum` int(11) unsigned NOT NULL DEFAULT '0',
  `rating_count` int(11) unsigned NOT NULL DEFAULT '0',
  `lastip` varchar(50) NOT NULL DEFAULT '',
  PRIMARY KEY (`content_id`)
)
ENGINE=MyISAM DEFAULT CHARSET=utf8;


-- Estructura de tabla para la tabla `jos_groups`
-- (Tabla de Grupos del Sitio)


CREATE TABLE IF NOT EXISTS `jos_groups`
(
  `id` tinyint(3) unsigned NOT NULL DEFAULT '0',
```

```
  `name` varchar(50) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
)
ENGINE=MyISAM DEFAULT CHARSET=utf8;

--
-- Volcar la base de datos para la tabla `jos_groups`
--

INSERT INTO `jos_groups` (`id`, `name`) VALUES
(0, 'Public'),
(1, 'Registered'),
(2, 'Special');



--
-- Estructura de tabla para la tabla `jos_hwdvidscategories`
-- (Categorías de Videos)


CREATE TABLE IF NOT EXISTS `jos_hwdvidscategories`
(
  `id` int(50) NOT NULL AUTO_INCREMENT,
  `parent` int(50) NOT NULL DEFAULT '0',
  `category_name` varchar(250) DEFAULT NULL,
  `category_description` text,
  `date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `access_b_v` tinyint(1) NOT NULL DEFAULT '0',
  `access_u_r` varchar(7) NOT NULL DEFAULT 'RECURSE',
  `access_v_r` varchar(7) NOT NULL DEFAULT 'RECURSE',
```

```
  `access_u` int(11) NOT NULL DEFAULT '-2',
  `access_lev_u` varchar(250) NOT NULL DEFAULT '0,1',
  `access_v` int(11) NOT NULL DEFAULT '-2',
  `access_lev_v` varchar(250) NOT NULL DEFAULT '0,1',
  `thumbnail` text NOT NULL,
  `num_vids` int(50) NOT NULL DEFAULT '0',
  `num_subcats` int(50) NOT NULL DEFAULT '0',
  `order_by` varchar(15) NOT NULL DEFAULT '0',
  `ordering` int(50) NOT NULL DEFAULT '0',
  `checked_out` int(11) NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `published` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=12 ;
```

-- Estructura de tabla para la tabla `jos_jcomments`

-- (Tabla de comentarios del sitio)

```
CREATE TABLE IF NOT EXISTS `jos_jcomments`
(
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `parent` int(11) unsigned NOT NULL DEFAULT '0',
  `path` varchar(255) NOT NULL DEFAULT '',
  `level` tinyint(1) unsigned NOT NULL DEFAULT '0',
  `object_id` int(11) unsigned NOT NULL DEFAULT '0',
  `object_group` varchar(255) NOT NULL DEFAULT '',
  `object_params` text NOT NULL,
```

```
 `lang` varchar(255) NOT NULL DEFAULT '',
 `userid` int(11) unsigned NOT NULL DEFAULT '0',
 `name` varchar(255) NOT NULL DEFAULT '',
 `username` varchar(255) NOT NULL DEFAULT '',
 `email` varchar(255) NOT NULL DEFAULT '',
 `homepage` varchar(255) NOT NULL DEFAULT '',
 `title` varchar(255) NOT NULL DEFAULT '',
 `comment` text NOT NULL,
 `ip` varchar(15) NOT NULL DEFAULT '',
 `date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
 `isgood` smallint(5) unsigned NOT NULL DEFAULT '0',
 `ispoor` smallint(5) unsigned NOT NULL DEFAULT '0',
 `published` tinyint(1) unsigned NOT NULL DEFAULT '0',
 `subscribe` tinyint(1) unsigned NOT NULL DEFAULT '0',
 `source` varchar(255) NOT NULL DEFAULT '',
 `source_id` int(11) unsigned NOT NULL DEFAULT '0',
 `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
 `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
 `editor` varchar(50) NOT NULL DEFAULT '',
 PRIMARY KEY (`id`),
 KEY `idx_userid` (`userid`),
 KEY `idx_source` (`source`),
 KEY `idx_email` (`email`),
 KEY `idx_lang` (`lang`),
 KEY `idx_subscribe` (`subscribe`),
 KEY `idx_checkout` (`checked_out`),
 KEY `idx_object` (`object_id`,`object_group`,`published`,`date`),
 KEY `idx_path` (`path`,`level`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=8 ;
```

-- Estructura de tabla para la tabla `jos_menu`

-- (Tabla de Menus)

```
CREATE TABLE IF NOT EXISTS `jos_menu`
(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `menutype` varchar(75) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `alias` varchar(255) NOT NULL DEFAULT '',
  `link` text,
  `type` varchar(50) NOT NULL DEFAULT '',
  `published` tinyint(1) NOT NULL DEFAULT '0',
  `parent` int(11) unsigned NOT NULL DEFAULT '0',
  `componentid` int(11) unsigned NOT NULL DEFAULT '0',
  `sublevel` int(11) DEFAULT '0',
  `ordering` int(11) DEFAULT '0',
  `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `pollid` int(11) NOT NULL DEFAULT '0',
  `browserNav` tinyint(4) DEFAULT '0',
  `access` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `utaccess` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `params` text NOT NULL,
  `lft` int(11) unsigned NOT NULL DEFAULT '0',
  `rgt` int(11) unsigned NOT NULL DEFAULT '0',
  `home` int(1) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `componentid` (`componentid`,`menutype`,`published`,`access`),
  KEY `menutype` (`menutype`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=111 ;
```

-- Estructura de tabla para la tabla `jos_menu_types`

-- (Tabla de Tipos de Menús)

```
CREATE TABLE IF NOT EXISTS `jos_menu_types`
(
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `menutype` varchar(75) NOT NULL DEFAULT '',
  `title` varchar(255) NOT NULL DEFAULT '',
  `description` varchar(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`),
  UNIQUE KEY `menutype` (`menutype`)
)
 ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;

--
-- Volcar la base de datos para la tabla `jos_menu_types`
--

INSERT INTO `jos_menu_types` (`id`, `menutype`, `title`, `description`) VALUES
(1, 'mainmenu', 'Main Menu', 'The main menu for the site'),
(2, 'usermenu', 'User Menu', 'A Menu for logged in Users'),
(3, 'topmenu', 'Top Menu', 'Top level navigation'),
(4, 'othermenu', 'Resources', 'Additional links'),
(5, 'ExamplePages', 'Example Pages', 'Example Pages'),
(6, 'keyconcepts', 'Key Concepts', 'This describes some critical information for new Users.');
```

--(Tabla de Módulos)

```
CREATE TABLE IF NOT EXISTS `jos_modules`
(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` text NOT NULL,
  `content` text NOT NULL,
  `ordering` int(11) NOT NULL DEFAULT '0',
  `position` varchar(50) DEFAULT NULL,
  `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `published` tinyint(1) NOT NULL DEFAULT '0',
  `module` varchar(50) DEFAULT NULL,
  `numnews` int(11) NOT NULL DEFAULT '0',
  `access` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `showtitle` tinyint(3) unsigned NOT NULL DEFAULT '1',
  `params` text NOT NULL,
  `iscore` tinyint(4) NOT NULL DEFAULT '0',
  `client_id` tinyint(4) NOT NULL DEFAULT '0',
  `control` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `published` (`published`,`access`),
  KEY `newsfeeds` (`module`,`published`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=138 ;
```

-- Estructura de tabla para la tabla `jos_phocagallery`

-- (Tabla de Información de Fotografías)

```
CREATE TABLE IF NOT EXISTS `jos_phocagallery`
```

```
(
 `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
 `catid` int(11) NOT NULL DEFAULT '0',
 `sid` int(11) NOT NULL DEFAULT '0',
 `title` varchar(250) NOT NULL DEFAULT '',
 `alias` varchar(255) NOT NULL DEFAULT '',
 `filename` varchar(250) NOT NULL DEFAULT '',
 `description` text,
 `date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
 `hits` int(11) NOT NULL DEFAULT '0',
 `latitude` varchar(20) NOT NULL DEFAULT '',
 `longitude` varchar(20) NOT NULL DEFAULT '',
 `zoom` int(3) NOT NULL DEFAULT '0',
 `geotitle` varchar(255) NOT NULL DEFAULT '',
 `videocode` text,
 `vmproductid` int(11) NOT NULL DEFAULT '0',
 `imgorigsize` int(11) NOT NULL DEFAULT '0',
 `published` tinyint(1) NOT NULL DEFAULT '0',
 `approved` tinyint(1) NOT NULL DEFAULT '0',
 `checked_out` int(11) NOT NULL DEFAULT '0',
 `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
 `ordering` int(11) NOT NULL DEFAULT '0',
 `params` text,
 `metakey` text,
 `metadesc` text,
 `extlink1` text,
 `extlink2` text,
 `extid` varchar(255) NOT NULL DEFAULT '',
 `extl` varchar(255) NOT NULL DEFAULT '',
 `extm` varchar(255) NOT NULL DEFAULT '',
```

```
  `exts` varchar(255) NOT NULL DEFAULT '',
  `exto` varchar(255) NOT NULL DEFAULT '',
  `extw` varchar(255) NOT NULL DEFAULT '',
  `exth` varchar(255) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`),
  KEY `catid` (`catid`,`published`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=13 ;
```

-- Estructura de tabla para la tabla `jos_polls`

-- (Tabla de Información de la encuesta)

```
CREATE TABLE IF NOT EXISTS `jos_polls`
(
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL DEFAULT '',
  `alias` varchar(255) NOT NULL DEFAULT '',
  `voters` int(9) NOT NULL DEFAULT '0',
  `checked_out` int(11) NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `published` tinyint(1) NOT NULL DEFAULT '0',
  `access` int(11) NOT NULL DEFAULT '0',
  `lag` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=16 ;
--
-- Volcar la base de datos para la tabla `jos_polls`
```

```
--

INSERT INTO `jos_polls` (`id`, `title`, `alias`, `voters`, `checked_out`, `checked_out_time`, `published`, `access`, `lag`) VALUES
(15, 'Â¿Que opina sobre nuestro nuevo sitio web?', 'ique-opina-sobre-nuestro-nuevo-sitio-web', 20, 0, '0000-00-00 00:00:00', 1, 0, 86400);


-- Estructura de tabla para la tabla `jos_poll_data`
-- (Tabla de Preguntas de Encuestas)


CREATE TABLE IF NOT EXISTS `jos_poll_data`
(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `pollid` int(11) NOT NULL DEFAULT '0',
  `text` text NOT NULL,
  `hits` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `pollid` (`pollid`,`text`(1))
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=25 ;


--
-- Volcar la base de datos para la tabla `jos_poll_data`
--

INSERT INTO `jos_poll_data` (`id`, `pollid`, `text`, `hits`) VALUES
(13, 15, 'Interesante ', 13),
```

```
(14, 15, 'Muy Buena ', 1),
(15, 15, 'Buena', 0),
(16, 15, 'Regular', 4),
(17, 15, '', 1),
(18, 15, '', 1),
(19, 15, '', 0),
(20, 15, '', 0),
(21, 15, '', 0),
(22, 15, '', 0),
(23, 15, '', 0),
(24, 15, '', 0);
```

-- Estructura de tabla para la tabla `jos_poll_date`

-- (Tabla de respuestas de encuestas)

```
CREATE TABLE IF NOT EXISTS `jos_poll_date`
(
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `vote_id` int(11) NOT NULL DEFAULT '0',
  `poll_id` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `poll_id` (`poll_id`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=34 ;
```

-- Estructura de tabla para la tabla `jos_sections`

-- (Tabla de secciones)

```
CREATE TABLE IF NOT EXISTS `jos_sections`
(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL DEFAULT '',
  `name` varchar(255) NOT NULL DEFAULT '',
  `alias` varchar(255) NOT NULL DEFAULT '',
  `image` text NOT NULL,
  `scope` varchar(50) NOT NULL DEFAULT '',
  `image_position` varchar(30) NOT NULL DEFAULT '',
  `description` text NOT NULL,
  `published` tinyint(1) NOT NULL DEFAULT '0',
  `checked_out` int(11) unsigned NOT NULL DEFAULT '0',
  `checked_out_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `ordering` int(11) NOT NULL DEFAULT '0',
  `access` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `count` int(11) NOT NULL DEFAULT '0',
  `params` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_scope` (`scope`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=10 ;
```

-- Estructura de tabla para la tabla `jos_session`

-- (Tabla de Sesiones)

```
CREATE TABLE IF NOT EXISTS `jos_session`
 (
  `username` varchar(150) DEFAULT '',
  `time` varchar(14) DEFAULT '',
  `session_id` varchar(200) NOT NULL DEFAULT '0',
  `guest` tinyint(4) DEFAULT '1',
  `userid` int(11) DEFAULT '0',
  `usertype` varchar(50) DEFAULT '',
  `gid` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `client_id` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `data` longtext,
  PRIMARY KEY (`session_id`(64)),
  KEY `whosonline` (`guest`,`usertype`),
  KEY `userid` (`userid`),
  KEY `time` (`time`)
)
ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

-- Estructura de tabla para la tabla `jos_users`

-- Creaciones de Usuarios Joomla

```
CREATE TABLE IF NOT EXISTS `jos_users`
(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL DEFAULT '',
  `username` varchar(150) NOT NULL DEFAULT '',
  `email` varchar(100) NOT NULL DEFAULT '',
  `password` varchar(100) NOT NULL DEFAULT '',
```

```
  `usertype` varchar(25) NOT NULL DEFAULT '',
  `block` tinyint(4) NOT NULL DEFAULT '0',
  `sendEmail` tinyint(4) DEFAULT '0',
  `gid` tinyint(3) unsigned NOT NULL DEFAULT '1',
  `registerDate` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `lastvisitDate` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `activation` varchar(100) NOT NULL DEFAULT '',
  `params` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `usertype` (`usertype`),
  KEY `idx_name` (`name`),
  KEY `gid_block` (`gid`,`block`),
  KEY `username` (`username`),
  KEY `email` (`email`)
)
ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=83 ;
```

#Esta Base de datos, no maneja claves foráneas ni constraints, funciones, ni #stored procedures.

1.12 CODIFICACIÓN DE COMPONENTES

C:\wamp\www\mashup\modules\mod_cblogin\mod_cblogin.php

```
<?php
/**
```

```
* Community builder Login Module 1.2 RC 3

* $Id: mod_cblogin.php 1079 2010-06-10 23:59:41Z beat $

*

* @version 1.2

* @package Community Builder 1.2 extensions

* @copyright (C) 2004-2010 Beat & JoomlaJoe & parts 2000 - 2005 Miro International Pty Ltd

* @license http://www.gnu.org/licenses/old-licenses/gpl-2.0.html GNU/GPL version 2

*

* Credits to: Jeffrey Randall for initial implementation of avatar, and

* to Antony Ventouris for the PMS integration (he also added the cool animated image)

*/

if ( ! ( defined( '_VALID_CB' ) || defined( '_JEXEC' ) || defined( '_VALID_MOS' ) ) ) { die( 'Direct Access to this location is not allowed.' ); }

/**

 * CB framework

 * @global CBframework $_CB_framework

 */

global $_CB_framework, $_CB_database, $ueConfig, $mainframe, $_SERVER;
```

```
if ( defined( 'JPATH_ADMINISTRATOR' ) ) {
	if ( ! file_exists( JPATH_ADMINISTRATOR .
'/components/com_comprofiler/plugin.foundation.php' ) ) {
		echo 'CB not installed';
		return;
	}
	include_once( JPATH_ADMINISTRATOR .
'/components/com_comprofiler/plugin.foundation.php' );
} else {
	if ( ! file_exists( $mainframe->getCfg( 'absolute_path' ) .
'/administrator/components/com_comprofiler/plugin.foundation.php' ) ) {
		echo 'CB not installed';
		return;
	}
	include_once( $mainframe->getCfg( 'absolute_path' ) .
'/administrator/components/com_comprofiler/plugin.foundation.php' );
}
cbimport( 'cb.database' );
cbimport( 'language.front' );
$absolute_path = $_CB_framework->getCfg( 'absolute_path' );
$cblogin_live_site = $_CB_framework->getCfg( 'live_site' );
```

```
$len_live_site =        strlen($cblogin_live_site);             // do not remove: used
further down as well

$isHttps        =       (isset($_SERVER['HTTPS']) && ( !empty(
$_SERVER['HTTPS'] ) ) && ($_SERVER['HTTPS'] != 'off') );

$return=        'http' . ( $isHttps ? 's' : '' ) . '://' . $_SERVER['HTTP_HOST'];

if (!empty ($_SERVER['PHP_SELF']) && ! empty ($_SERVER['REQUEST_URI']))
{

        $return=        $_SERVER['REQUEST_URI'];        // Apache

} else {

        $return.=       $_SERVER['SCRIPT_NAME'];        // IIS

        if (isset($_SERVER['QUERY_STRING']) && !
empty($_SERVER['QUERY_STRING'])) {

                $return.=       '?' . $_SERVER['QUERY_STRING'];

        }

}

$return=        preg_replace('/[\\\"\\\'][\\s]*javascript:(.*)[\\\"\\\']/', """",
preg_replace('/eval\((.*)\)/', '', htmlspecialchars( urldecode( $return ) ) ) );


$return = cbUnHtmlspecialchars( $return );

// avoid unauthorized page acces at very first login after registration confirmation
```

```
if (preg_match(
'/index.php\?option=com_comprofiler&task=confirm&confirmCode=|index.php\?opti
on=com_comprofiler&task=login/', $return)) $return = "index.php";


// $params is defined by include: ignore this warning:
if (is_callable(array($params,"get"))) {                    // Mambo 4.5.0 compatibility
        $message_login      = $params->get( 'login_message', 0 );
        $message_logout = $params->get( 'logout_message', 0 );
        $pretext            = $params->get( 'pretext' );
        $posttext           = $params->get( 'posttext' );
        $logoutpretext      = $params->get( 'logoutpretext' );
        $logoutposttext = $params->get( 'logoutposttext' );
        $login              = $params->get( 'login', $return );
        $logout             = $params->get( 'logout', "index.php" );
        if ( $logout == '#' ) {
                $logout             = $return;
        }
        $name               = $params->get( 'name', 0 );
        $greeting           = $params->get( 'greeting', 1 );
        $class_sfx          = $params->get( 'moduleclass_sfx', "");
        $horizontal         = $params->get( 'horizontal', 0);
```

```
        $show_avatar = $params->get( 'show_avatar', 0);
        $avatar_position = $params->get( 'avatar_position', "default");
        $text_show_profile = $params->get( 'text_show_profile', "");
        $text_edit_profile = $params->get( 'text_edit_profile', "");
        $pms_type          = $params->get( 'pms_type', 0);
        $show_pms          = $params->get( 'show_pms', 0);
        $remember_enabled = $params->get( 'remember_enabled', 1);
        $https_post        = $params->get( 'https_post', 0);
        $showPendingConnections = $params->get( 'show_connection_notifications',
0);
$show_newaccount = $params->get( 'show_newaccount', 1 );
$show_lostpass     = $params->get( 'show_lostpass', 1 );
$name_lenght       = $params->get( 'name_lenght', "14" );
$pass_lenght   = $params->get( 'pass_lenght', "14" );
$compact           = $params->get( 'compact', 0 );
$cb_plugins        = $params->get( 'cb_plugins', 0 );
$show_username_pass_icons=      $params->get( 'show_username_pass_icons', 0 );
$show_buttons_icons             =       $params->get( 'show_buttons_icons', 0 );
$show_remind_register_icons     =       $params->get(
'show_remind_register_icons', 0 );
} else {
```

```
$message_login        = 0;
$message_logout = 0;
$pretext              = "";
$posttext             = "";
$logoutpretext= "";
$logoutposttext       = "";
$login                = $return;
$logout               = "index.php";
$name                 = 0;
$greeting             = 1;
$class_sfx            = "";
$horizontal           = 0;
$show_avatar  = 0;
$avatar_position = "default";
$text_show_profile = "";
$text_edit_profile = "";
$pms_type             = 0;
$show_pms             = 0;
$remember_enabled = 1;
$https_post           = 0;
```

```
        $showPendingConnections = 0;
        $show_newaccount = 1;
        $show_lostpass          = 1;
        $name_lenght            = "10";
        $pass_lenght   = "10";
        $compact                = 0;
        $cb_plugins             = true;
        $show_username_pass_icons=      0;
}
$id_sfx     =   trim( preg_replace( '/\s+/', '_', $class_sfx ) );
switch ( checkJversion() ) {
        case 0:
        // Mambo 4.5 & Joomla 1.0:
                $urlImgPath = $cblogin_live_site."/modules/mod_cblogin/";
                break;
        case -1:
        // Mambo 4.6.x:
                $urlImgPath = $cblogin_live_site."/modules/mod_cblogin/";
                break;
        case 1:
```

```
        default:
        // Joomla 1.5+
                $urlImgPath =
$cblogin_live_site."/modules/mod_cblogin/mod_cblogin/";
                $lang =& JFactory::getLanguage();
                $lang->load("mod_login");               // might not be needed, unsure
                break;
}
if ( $_CB_framework->myId() ) {
        if ($name) {
                if ($name == 2) {
                        $query = "SELECT firstname FROM #__comprofiler WHERE
id = ". (int) $_CB_framework->myId();
                } else {
                        $query = "SELECT name FROM #__users WHERE id = ".
(int) $_CB_framework->myId();
                }
                $_CB_database->setQuery( $query );
                // some joomla versions (1.5.1, 1.0.11+ do htmlspecialchars in
database!):
                $name = htmlspecialchars( cbUnHtmlspecialchars( $_CB_database-
>loadResult() ) );
```

```
	} else {
		$name = htmlspecialchars( cbUnHtmlspecialchars( $_CB_framework->myUsername() ) );
	}
	$logout = cbSef( htmlspecialchars( $logout ) );
	// wondering if this was even neeeded: echo '<div style="width:100%; padding:0px; border-width:0px; margin:0px;">'."\n";
	$logoutPost = $_CB_framework->viewUrl( 'logout' );
	echo '<form action="'.$logoutPost.'" method="post" id="mod_login_logoutform'.$id_sfx.'" style="margin:0px;">'."\n";
	// Add Authenticated Pre text
	if ( $logoutpretext ) {
		if ( defined( $logoutpretext ) ) {
			echo constant( $logoutpretext );
		} else {
			echo $logoutpretext;
		}
		echo "\n";
	}
	$avatarDisplayed = false;
	if ($show_avatar == 0) {
```

```
if ($greeting) echo '<span id="mod_login_greeting'.$id_sfx.'">'.sprintf( _UE_HI_NAME, $name ).'</span>'.($horizontal ? " " : "");

} else {

if (($avatar_position=="default") and ($greeting)) echo '<span id="mod_login_greeting'.$id_sfx.'">'.sprintf( _UE_HI_NAME, $name ).'</span>';


$cbUser =& CBuser::getInstance( $_CB_framework->myId() );

$oValue = $cbUser->avatarFilePath( $show_avatar );

if ($oValue) {

if ($avatar_position=="default") echo '<div style="text-align:center; margin-left:auto; margin-right:auto;">';

else echo '<div style="float:'.$avatar_position.'; margin: 3px 0px 4px 0px; ">';

echo '<a href="' . $_CB_framework->userProfileUrl() . '" class="mod_login'.$class_sfx.'">';

echo '<img src="' . htmlspecialchars( $oValue ) . '" style="margin: 0px 1px 3px 1px; border-width:0px;" alt="'.$name

. '" title="'. _UE_MENU_VIEWMYPROFILE_DESC . '" class="mod_login'.$class_sfx.'" id="mod_login_avatar'.$id_sfx.'" />';

echo "</a></div>\n";
```

```
                $avatarDisplayed = true;
            }
        }
            if ( trim( $text_show_profile ) ) {
            if ( ! ( ( $compact && ( $show_remind_register_icons == 2 ) ) || $horizontal ) ) {
                if ($avatar_position=="default") {
                    echo '<div style="text-align:center; margin-left:auto; margin-right:auto;">';
                } else {
                    echo '<div style="float:'.$avatar_position.'; margin: 3px 0px 4px 0px; ">';
                }
            }
            echo '<a href="' . $_CB_framework->userProfileUrl() . '" class="mod_login'.$class_sfx.'">';
            echo '<span title="' . _UE_MENU_VIEWMYPROFILE_DESC . '" class="mod_login_showprofile'.$class_sfx.'">';
            if ( $show_remind_register_icons >= 1 ) {
                echo '<img src="' . $urlImgPath . 'username.png" alt="' . _UE_MENU_VIEWMYPROFILE_DESC . '" width="25px" height="20px" style="border-width:0px;cursor:pointer;" /> ';
```

```
                }
                if ( $show_remind_register_icons <= 1 ) {
                        if ( defined( $text_show_profile ) ) {
                                echo constant( $text_show_profile );
                        } else {
                                echo $text_show_profile;
                        }
                }
                echo '</span>';
                echo '</a>';
                if ( ! ( ( $compact && ( $show_remind_register_icons == 2 ) ) ||
$horizontal ) ) {
                        echo "</div>\n";
                }
        }
        if ( trim( $text_edit_profile ) ) {
                if ( ! ( ( $compact && ( $show_remind_register_icons == 2 ) ) ||
$horizontal ) ) {
                        if ($avatar_position=="default") {
                                echo '<div style="text-align:center; margin-left:auto;
margin-right:auto;">';
```

```
                } else {
                        echo '<div style="float:'.$avatar_position.'; margin: 3px
0px 4px 0px; ">';
                }
        }
        echo '<a href="' . $_CB_framework->userProfileEditUrl() . '"
class="mod_login'.$class_sfx.'">';
        echo '<span title="' . _UE_EDIT_TITLE . '"
class="mod_login_editprofile'.$class_sfx.'">';
        if ( $show_remind_register_icons >= 1 ) {
                echo '<img src="' . $urlImgPath . 'edit_profile.png" alt="' .
_UE_EDIT_TITLE . '" width="25px" height="20px" style="border-
width:0px;cursor:pointer;" /> ';
        }
        if ( $show_remind_register_icons <= 1 ) {
                if ( defined( $text_edit_profile ) ) {
                        echo constant( $text_edit_profile );
                } else {
                        echo $text_edit_profile;
                }
        }
        echo '</span>';
```

```
                echo '</a>';
                if ( ! ( ( $compact && ( $show_remind_register_icons == 2 ) ) ||
$horizontal ) ) {
                        echo "</div>\n";
                }
        }
        if ( !$horizontal ) {
                $preDiv = '<div style="text-align:center; margin:0px auto;"> '
                        . '<div style="margin:auto; align:center; width:100%;">
'
                        . '<div style="display:table; margin:auto; align:center;';
                $postDiv = "</div></div></div>\n";
        }

        if ( $show_avatar and ($avatar_position!="default") and ($greeting) ) {
                if ($avatarDisplayed && ( ! $horizontal ) ) {
                        echo $preDiv . '" id="mod_login_greeting'.$id_sfx.'">';
                        echo '<br />';
                        echo sprintf( _UE_HI_NAME, '<br />'.$name );
                        echo $postDiv;
                } else {
```

```
                        echo '<span id="mod_login_greeting'.$id_sfx.'">'.sprintf(
_UE_HI_NAME, $name ).'</span>';
                }
        }
                $pms = 0;
        if($show_pms != 0) {
                $pms = $pms_type;               // RC2 quick fix
                if($pms != 0)
                {
                        switch ($pms) {
                                case 1:
                                        $pmsnameprefix = "";
                                        $query_pms_count = "SELECT count(id)
FROM #__".$pmsnameprefix."pms WHERE username=" . $_CB_database->Quote(
$_CB_framework->myUsername() ) . " AND readstate=0";
                                        $_CB_database->setQuery( $query_pms_count
);
                                        $total_pms = $_CB_database->loadResult();
                                        $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE
'%com_".$pmsnameprefix."pms%'";
                                        $_CB_database->setQuery( $query_pms_link );
```

```
                $pms_link_id = $_CB_database->loadResult();

                $pms_link =
"index.php?option=com_".$pmsnameprefix."pms&amp;page=index".($pms_link_id
? "&amp;Itemid=".$pms_link_id : "");

                break;

            case 2:

                $pmsnameprefix = "my";

                $query_pms_count = "SELECT count(id)
FROM #__".$pmsnameprefix."pms WHERE username=" . $_CB_database->Quote(
$_CB_framework->myUsername() ) . " AND readstate=0";

                $_CB_database->setQuery( $query_pms_count
);

                $total_pms = $_CB_database->loadResult();

                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE
'%com_".$pmsnameprefix."pms%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();

                $pms_link =
"index.php?option=com_".$pmsnameprefix."pms&amp;task=inbox".($pms_link_id ?
"&amp;Itemid=".$pms_link_id : "");

                break;

            case 3:
```

```
                $query_pms_count="SELECT count(u.id)
FROM #__uddeim AS u WHERE u.totrash=0 AND u.toread=0 AND u.toid=" . (int)
$_CB_framework->myId();

                $_CB_database->setQuery($query_pms_count);

                $total_pms = $_CB_database->loadResult();


                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE '%com_uddeim%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();

                $pms_link =
"index.php?option=com_uddeim&amp;task=inbox".($pms_link_id ?
"&amp;Itemid=".$pms_link_id : "");

                break;

            case 4:         // PMS Enhanced by Stefan:

                $pmsnameprefix = "";

                $query_pms_count = "SELECT count(id)
FROM #__".$pmsnameprefix."pms WHERE username=" . $_CB_database->Quote(
$_CB_framework->myUsername() ) . " AND readstate=0 AND inbox=1";

                $_CB_database->setQuery( $query_pms_count
);

                $total_pms = $_CB_database->loadResult();
```

```
                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE
'%com_".$pmsnameprefix."pms%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();

                $pms_link =
"index.php?option=com_".$pmsnameprefix."pms&amp;page=inbox".($pms_link_id
? "&amp;Itemid=".$pms_link_id : "");

                break;

        case 5:         // Clexus:

                $pmsnameprefix = "my";

                $query_pms_count = "SELECT count(id)
FROM #__".$pmsnameprefix."pms WHERE userid='" . (int) $_CB_framework-
>myId() . "' AND readstate=0";

                $_CB_database->setQuery( $query_pms_count
);

                $total_pms = $_CB_database->loadResult();


                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE
'%com_".$pmsnameprefix."pms%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();
```

```
                $pms_link =
"index.php?option=com_".$pmsnameprefix."pms&amp;task=inbox".($pms_link_id ?
"&amp;Itemid=".$pms_link_id : "");

                break;

        case 6:         // PMS Enhanced 2.x by Stefan:

                $pmsnameprefix = "";

                $query_pms_count = "SELECT count(id)
FROM #__".$pmsnameprefix."pms WHERE recip_id=" . (int) $_CB_framework-
>myId() . " AND readstate%2=0 AND inbox=1";

                $_CB_database->setQuery( $query_pms_count
);

                $total_pms = $_CB_database->loadResult();


                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE
'%com_".$pmsnameprefix."pms%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();

                $pms_link =
"index.php?option=com_".$pmsnameprefix."pms&amp;page=inbox".($pms_link_id
? "&amp;Itemid=".$pms_link_id : "");

                break;

        case 7:
```

```
                $pmsnameprefix="missus";

            $query_pms_count = "SELECT COUNT(*) FROM #__missus AS m
JOIN #__missus_receipt AS r WHERE m.id=r.id AND r.receptorid='" . (int)
$_CB_framework->myId() . "' AND r.rptr_rstate=0 AND r.rptr_tstate=0 AND
r.rptr_dstate=0 AND m.is_draft=0";

            $_CB_database->setQuery( $query_pms_count );

            $total_pms = $_CB_database->loadResult();


            $query_pms_link = "SELECT id FROM #__menu WHERE
published>=0 AND link LIKE '%com_".$pmsnameprefix."%'";

            $_CB_database->setQuery( $query_pms_link );

            $pms_link_id = $_CB_database->loadResult();

            $pms_link =
"index.php?option=com_".$pmsnameprefix."&amp;func=showinbox".($pms_link_id
? "&amp;Itemid=".$pms_link_id : "");

            break;

        case 8:

            $pmsnameprefix="jim";

            $query_pms_count = "SELECT COUNT(id) FROM #__jim WHERE
username=" . $_CB_database->Quote( $_CB_framework->myUsername() ) . " AND
readstate=0";

            $_CB_database->setQuery( $query_pms_count );

            $total_pms = intval($_CB_database->loadResult());
```

```
                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE '%com_".$pmsnameprefix."%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();

                $pms_link =
"index.php?option=com_".$pmsnameprefix.($pms_link_id ?
"&amp;Itemid=".$pms_link_id : "");

                break;

            case 9:

                $pmsnameprefix="primezilla";

                $query_pms_count = "SELECT COUNT(*)
FROM #__primezilla_inbox WHERE userid=" . (int) $_CB_framework->myId() . "
AND flag_read=0 AND flag_deleted=0";

                $_CB_database->setQuery( $query_pms_count
);

                $total_pms = intval($_CB_database-
>loadResult());

                $query_pms_link = "SELECT id FROM
#__menu WHERE published>=0 AND link LIKE '%com_".$pmsnameprefix."%'";

                $_CB_database->setQuery( $query_pms_link );

                $pms_link_id = $_CB_database->loadResult();

$pms_link = "index.php?option=com_".$pmsnameprefix.($pms_link_id ?
"&Itemid=".$pms_link_id : "");
```

```
                        break;
        case 10:        // JAM (Joomla Advanced Message), J1.5 only:
                        // Amount unread messages:
                        $query = 'SELECT COUNT(id)'
                                . ' FROM `#__jam_receivers`'
. ' WHERE rid = ' . (int) $_CB_framework->myId() . ' AND inbox = 1 AND state =
0';
                $_CB_database->setQuery( $query );
        $total_pms      =       $_CB_database->loadResult();

                        // JAM url:
                        if ( checkJversion() == 1 ) {
                $menu =&        JSite::getMenu();
                $item           =       $menu->getItems( 'link',
'index.php?option=com_jam&view=inbox', true);
                        } else {
                                $item           =       false;
                        }
                        if ( $item ) {
        $pms_link       =       'index.php?Itemid=' . $item->id;
                        } else {
```

```
        $pms_link        =        'index.php?option=com_jam&view=inbox';
                                  }
                                  break;
/* Test-code for SMF PMS integration: to be validated with SMF team before
integration !
                          case xxx:
                                  global $user_info;
                                  $total_pms = $user_info['unread_messages'];
$pms_link = ???
*/
default:
                                  break;
                  }
                  $pmsMsg = "";
                  if (($total_pms) > 0 ) {
                          $pmsMsg .= '<a href="'.cbSef("$pms_link").'"
class="mod_login'.$class_sfx.'" id="mod_login_pmsimg'.$id_sfx.'">';
                          $pmsMsg .= '<img border="0"
src="'.$urlImgPath.'mail.gif" width="14" height="15" alt="NEW"
class="mod_login'.$class_sfx.'" id="mod_login_messagesimg'.$id_sfx.'" /></a><br
/>'."\n";
```

```
                $pmsMsg .= '<a href="'.cbSef("$pms_link").'"
class="mod_login'.$class_sfx.'" id="mod_login_pmsa'.$id_sfx.'">';

                $pmsMsg .= '<span
id="mod_login_messagestext'.$id_sfx.'">'._UE_PM_MESSAGES_HAVE."
".$total_pms." ".($total_pms == 1 ? _UE_PM_NEW_MESSAGE :
_UE_PM_NEW_MESSAGES)."</span></a>\n";

            } else {

                if($show_pms >= 2 ) {

                    $pmsMsg .= '<a href="'.cbSef("$pms_link").'"
class="mod_login'.$class_sfx.'" id="mod_login_no_pms'.$id_sfx.'">';

                    $pmsMsg .= '<span
id="mod_login_nomessagestext'.$id_sfx.'">'._UE_PM_NO_MESSAGES."</span></
a>\n";

                }

            }

            if ($pmsMsg) {

                if ( !$horizontal ) echo $preDiv.' margin-top:0.7em;"
id="mod_login_pms'.$id_sfx.'">';

                echo $pmsMsg;

                if ( !$horizontal ) echo $postDiv;

            }

        }

    }
```

```
if($showPendingConnections) {

        if(isset($ueConfig['allowConnections']) &&
$ueConfig['allowConnections']) {

            // $query = "SELECT count(*) FROM
#__comprofiler_members WHERE pending=1 AND memberid=". (int)
$_CB_framework->myId();

            $query = "SELECT COUNT(*)"

            . "\n FROM #__comprofiler_members AS m"

            . "\n LEFT JOIN #__comprofiler AS c ON m.referenceid=c.id"

            . "\n LEFT JOIN #__users AS u ON m.referenceid=u.id"

            . "\n WHERE m.memberid=" . (int) $_CB_framework->myId()
. " AND m.pending=1"

            . "\n AND c.approved=1 AND c.confirmed=1 AND
c.banned=0 AND u.block=0"

            ;

            if(!$_CB_database->setQuery($query)) print $_CB_database-
>getErrorMsg();

            $totalpendingconnections = $_CB_database->loadResult();

            if($totalpendingconnections > 0) {

                if ( !$horizontal ) echo '<div style="margin:0.7em 0px
0px 0px; align:center; text-align:center;" id="mod_login_connections'.$id_sfx.'">';

                echo "<span id='mod_login_pendingConn".$id_sfx."'>";
```

```
                                    echo "<a href='" . $_CB_framework->viewUrl(
'manageconnections' ) . "' class='mod_login".$class_sfx."'
id='mod_login_connectimg".$id_sfx."'>";

                                    echo '<img border="0" src="'.$urlImgPath.'users.gif"
width="21" height="15" alt="NEW" class="mod_login'.$class_sfx.'"
id="mod_login_connections_img'.$id_sfx.'" />';

                                    echo "</a> ";

                                    echo "<a href='" . $_CB_framework->viewUrl(
'manageconnections' ) . "' class='mod_login".$class_sfx."'
id='mod_login_connect".$id_sfx."'>";

                                    echo _UE_PM_MESSAGES_HAVE."
".$totalpendingconnections." "._UE_CONNECTIONREQUIREACTION."</a
></span>";

                                    if ( !$horizontal ) echo "</div>";

                            }

                    }

            }

            if (!$horizontal) {

                    if ((!$avatarDisplayed) or ($avatar_position!="default") or ($pms))
$topMargin = "1.4em";

                    else $topMargin = "2px";

                    echo '<div style="text-align:center; margin:auto; margin:
'.$topMargin.' 0px 2px 0px;">';
```

```
	}
	if ( $cb_plugins ) {
		include_once( $absolute_path .
"/administrator/components/com_comprofiler/plugin.class.php");
		global $_PLUGINS;

		$_PLUGINS->loadPluginGroup('user');
		$pluginsResults = $_PLUGINS->trigger(
'onAfterLogoutForm', array( $name_lenght, $pass_lenght, $horizontal, $class_sfx,
&$params ) );
		if ( implode( $pluginsResults ) != '' ) {
			$divHtml = ( $horizontal ? '<span
class="mod_logout_plugin'.$class_sfx.'">' : '<div
class="mod_logout_plugin'.$class_sfx.'">' );
			$divHtmlEnd = ( $horizontal ? '</span>' : '</div>' );
			echo $divHtml . implode( $divHtmlEnd . $divHtml,
$pluginsResults ) . $divHtmlEnd;
		}
	}
	// Logout button/icon:
	switch ( $show_buttons_icons ) {
		case 2:
```

```
                $buttonStyle  =  ' style="width:25px;height:20px;border-
width:0px;margin:0px;cursor:pointer;vertical-align:top;background-image:url(' .
$urlImgPath . 'logout.png);background-position:0 0;background-repeat:no-repeat;"'
                ' title="' . _UE_BUTTON_LOGOUT . '"';
                $buttonValue  =  '';
                break;
        case 1:
                $buttonStyle  =  ' style="min-height:20px;padding-
left:30px;cursor:pointer;background-image:url(' . $urlImgPath .
'logout.png);background-position:0 0;background-repeat:no-repeat;width:auto;"';
                $buttonValue  =  _UE_BUTTON_LOGOUT;
                break;
        case 0:
        default:
                $buttonStyle  =  '';
                $buttonValue  =  _UE_BUTTON_LOGOUT;
                break;
    }
    echo '<input type="submit" name="Submit" class="button'.$class_sfx.'"
value="' . $buttonValue . '" . $buttonStyle . ' />';
    echo "\n".'<input type="hidden" name="op2" value="logout" />'."\n";
```

```
        echo '<input type="hidden" name="lang" value="' . $_CB_framework-
>getCfg( 'lang' ) . '" />'."\n";

        echo '<input type="hidden" name="return" value="B:' . base64_encode(
$logout ) . '" />'."\n";

        echo '<input type="hidden" name="message" value="' . htmlspecialchars(
$message_logout ) . '" />'."\n";

        echo cbGetSpoofInputTag( 'logout' );

        // this is left for backwards compatibility only, to be removed after CB 1.2:

        if ( is_callable("josSpoofValue")) {

                $validate = josSpoofValue( 1 );

                echo "<input type=\"hidden\" name=\"" .  $validate . "\" value=\"1\"
/>\n";

        }

        if ( !$horizontal ) echo "</div>";

        echo "</form>";                    // wondering if this was even neeeded: </div>";

        // Add Authenticated Post text

        if ( $logoutposttext ) {

                if ( defined( $logoutposttext ) ) {

                        echo constant( $logoutposttext );

                } else {

                        echo $logoutposttext;
```

```
                }
                echo "\n";
        }
} else {        // Login Form :
        switch ( isset( $ueConfig['login_type'] ) ? $ueConfig['login_type'] : 0 ) {
                case 2:
                        $userNameText       =       _UE_EMAIL;
                        break;
                case 1:
        // NEXT 3 LINES: CB 1.2 RC 2 + CB 1.2 specific : remove after !
                        if ( ! defined( '_UE_USERNAME_OR_EMAIL' ) ) {
                                DEFINE('_UE_USERNAME_OR_EMAIL','Username
or email');
                        }
                        $userNameText       =       _UE_USERNAME_OR_EMAIL;
                        break;
                case 0:
                default:
                        $userNameText       =       _UE_USERNAME;
                        break;
```

```
	}

	// redirect to site url (so cookies are recognized correctly after login):

	if (strncasecmp($cblogin_live_site, "http://www.", 11)==0 // &&
strncasecmp($cblogin_live_site, "http://", 7)==0

		&& strncasecmp( substr($cblogin_live_site, 11), substr($login, 7),
$len_live_site - 11 ) == 0 ) {

			// the login return string matches the live site without 'www.' in
it:

			// add www subdomain as live_site has it.

			$login = "http://www." . substr($login, 7);

	} elseif (strncasecmp($cblogin_live_site, "https://www.", 12)==0 // &&
strncasecmp($cblogin_live_site, "https://", 8)==0

		&& strncasecmp( substr($cblogin_live_site, 12), substr($login, 8),
$len_live_site - 12 ) == 0 ) {

			$login = "https://www." . substr($login, 8);  // same for https

/* However, we can't remove www in joomla 1.0.13+, because cookies would fail on
domain test to allow for login:

	} elseif (strncasecmp($cblogin_live_site, "http://", 7)==0 &&
strncasecmp($cblogin_live_site, "http://www.", 11)==0

		&& strncasecmp( substr($cblogin_live_site, 7), substr($login, 11),
$len_live_site - 7 ) == 0 ) {

			$login = "http://" . substr($login, 11);
```

```
	} elseif (strncasecmp($cblogin_live_site, "https://", 8)==0 &&
strncasecmp($cblogin_live_site, "https://www.", 12)==0
		&& strncasecmp( substr($cblogin_live_site, 8), substr($login, 12),
$len_live_site - 8 ) == 0 ) {
			$login = "https://" . substr($login, 12);
*/
	}
	$login = cbSef( $login );
	if ( $https_post > 1 /* && ! $isHttps */ ) {
		if ((strncmp($login, "http:", 5)!=0) && (strncmp($login, "https:",
6)!=0)) {
			$login = $cblogin_live_site . '/' . $login;
		}
		$login = str_replace("http://","https://",$login);
	}
	$loginPost	=	$_CB_framework->viewUrl( 'login' );
	if ( $https_post /* && ! $isHttps */ ) {
		if ( ( substr($loginPost, 0, 5) != "http:" ) && ( substr($loginPost, 0, 6)
!= "https:" ) ) {
			$loginPost = $cblogin_live_site."/".$loginPost;
		}
```

```
            $loginPost = str_replace("http://","https://",$loginPost);
        }
        // now we need to make sure that the cookie in return of this post is sent to the most generic domain, in case multiple domains exist:
        // if the current page ($return) is without www, then login should also be without www, even if live_site has www:
        if (strncasecmp($loginPost, "http://www.", 11)==0 // && strncasecmp($cblogin_live_site, "http://", 7)==0
            && strncasecmp( substr($loginPost, 11), substr($return, 7), $len_live_site - 11 ) == 0 ) {
                // the login return string matches the live site without 'www.' in it:
                // add www subdomain as live_site has it.
                $loginPost = "http://" . substr($loginPost, 11);
        } elseif (strncasecmp($loginPost, "https://www.", 12)==0 // && strncasecmp($cblogin_live_site, "https://", 8)==0
            && strncasecmp( substr($loginPost, 12), substr($return, 8), $len_live_site - 12 ) == 0 ) {
                $loginPost = "https://" . substr($loginPost, 12);       // same for https
        }
        echo '<form action="'.$loginPost.'" method="post" id="mod_loginform'.$id_sfx.'" ';
```

```
/*
	if ($compact) {
		echo "onsubmit=\""
			. "return ( ( this.elements['mod_login_username" . $class_sfx .
"'].value != '"._USERNAME."')"
			. "&& ( this.elements['mod_login_password" . $class_sfx .
"'].value != 'paswww') )"
			."\" ";
	}
*/
	echo 'style="margin:0px;">'."\n";
	if ( $pretext ) {
		if ( defined( $pretext ) ) {
			echo constant( $pretext );
		} else {
			echo $pretext;
		}
		echo "\n";
	}
	if (!$horizontal) {
```

```
		echo '<table width="100%" border="0" cellspacing="0"
cellpadding="0" class="mod_login'.$class_sfx.'">'."\n"
		. "<tr><td>";
	}
	$bgstyleUser		=	'';
	$bgstylePass		=	'';
	if ( $compact ) {
		if ( $show_username_pass_icons >= 1 ) {
			$bgstyleUser  .=	' style="background-image:url(' .
$urlImgPath . 'username.png); background-repeat: no-repeat; background-position:
0px 0px; padding-left: 30px; min-height: 18px;width:auto;" ';
			$bgstylePass  .=	' style="background-image:url(' .
$urlImgPath . 'password.png); background-repeat: no-repeat; background-position:
0px 0px; padding-left: 30px; min-height: 18px;width:auto;" ';
		}
		if ( $show_username_pass_icons <= 1 ) {
			$bgstyleUser  .=	" alt=\"" . htmlspecialchars(
$userNameText ) . "\" value=\"" . htmlspecialchars( $userNameText ) . "\" "
. "onfocus=\"if
(this.value=='" . addslashes( $userNameText ) . "') this.value="\"
onblur=\"if(this.value==") { this.value='" . addslashes( $userNameText ) . "'; return
false; }\"";
```

```
                $bgstylePass .= " alt=\"".JUE_PASS."\"
value=\"paswww\" onfocus=\"if (this.value=='paswww') this.value="\"
onblur=\"if(this.value==") { this.value='paswww'; return false; }\"";

            }

        } else {

            if ( $show_username_pass_icons == 2 ) {

                $bgstyleUser .= ' style="vertical-align:top;" ';

                $bgstylePass .= ' style="vertical-align:top;" ';

            }

            $txtusername = '<label
for="mod_login_username'.$class_sfx.'">'

                                                . (
$show_username_pass_icons >= 1 ? '<img src="' . $urlImgPath . 'username.png"
width="25" height="20" alt="' . $userNameText . '" /> ' : " )

                                                . (
$show_username_pass_icons <= 1 ? $userNameText : " )

                                            . '</label>';

            $txtpassword = '<label
for="mod_login_password'.$class_sfx.'">'

                                                . (
$show_username_pass_icons >= 1 ? '<img src="' . $urlImgPath . 'password.png"
width="25" height="20" alt="' . JUE_PASS . '" /> ' : " )
```

```
                                                    .       (
$show_username_pass_icons <= 1 ? _UE_PASS : '' )
                                                    .       '</label>';
            echo '<span
id="mod_login_usernametext'.$id_sfx.'">'.$txtusername.'</span>';
        }
        if ( $compact || ( $show_username_pass_icons == 2 ) ) {
            $bgstyleUser        .=      ' title="' . $userNameText . '"';
            $bgstylePass        .=      ' title="' . _UE_PASS . '"';
        }
        if ($horizontal) {
            echo " \n";
        } elseif ( ( ! $compact ) && ( $show_username_pass_icons != 2 ) ) {
            echo "<br />\n";
        }
        echo '<input type="text" name="username"
id="mod_login_username'.$id_sfx.'" class="inputbox'.$class_sfx.'"
size="'.$name_lenght.'"' . $bgstyleUser . ' />';
        if ($horizontal) {
            echo " \n";
        } else {                                    // if (!$compact) {
```

```
            echo "<br />\n";
        }
        if (!$compact) {
            echo '<span
id="mod_login_passwordtext'.$id_sfx.'">'.$txtpassword.'</span>';
        }
        if ($horizontal) {
            echo " ";
        } elseif ( ( ! $compact ) && ( $show_username_pass_icons != 2 ) ) {
            echo "<br />";
        }
        echo '<span><input type="password" name="passwd"
id="mod_login_password'.$id_sfx.'" class="inputbox'.$class_sfx.'"
size="'.$pass_lenght."" . $bgstylePass . ' /></span>';
        if ($horizontal) {
            echo " \n";
        } else {
            echo "<br />\n";
        }
        echo '<input type="hidden" name="op2" value="login" />'."\n";
```

```
        echo '<input type="hidden" name="lang" value="' . $_CB_framework-
>getCfg( 'lang' ) . '" />' . "\n";

        echo '<input type="hidden" name="force_session" value="1" />'."\n";
        // makes sure to create joomla 1.0.11+12 session/bugfix

        echo '<input type="hidden" name="return" value="B:' . base64_encode(
$login ) . '" />'."\n";

        echo '<input type="hidden" name="message" value="' . htmlspecialchars(
$message_login ) . '" />'."\n";

        $loginFrom = ( defined( '_UE_LOGIN_FROM' ) ? constant(
'_UE_LOGIN_FROM' ) : 'loginmodule' );

        echo '<input type="hidden" name="loginfrom" value="' . htmlspecialchars(
$loginFrom ) . '" />'."\n";

        echo cbGetSpoofInputTag( 'login' );

// this is left for backwards compatibility only, to be removed after CB 1.2:

        if ( is_callable("josSpoofValue")) {

                $validate = josSpoofValue( 1 );

                echo "<input type=\"hidden\" name=\"" .  $validate . "\" value=\"1\"
/>\n";

        }

        switch ($remember_enabled) {

                case 1:
```

```
                    echo '<input type="checkbox" name="remember"
id="mod_login_remember'.$id_sfx.'" . ( $class_sfx ? ' class="inputbox'.$class_sfx.'"
: '' ) . ' value="yes" /> '

                            .'<span
id="mod_login_remembermetext'.$id_sfx.'"><label
for="mod_login_remember'.$class_sfx.'">'._UE_REMEMBER_ME."</label></span
>";

                    if ($horizontal) echo " \n"; else echo "<br />\n";

                break;

                case 2:

                    echo '<input type="hidden" name="remember" value="yes" />';

                break;

                case 3:

                    echo '<input type="checkbox" name="remember"
id="mod_login_remember'.$id_sfx.'" . ( $class_sfx ? ' class="inputbox'.$class_sfx.'"
: '' ) . ' value="yes" checked="checked" /> '

                            .'<span
id="mod_login_remembermetext'.$id_sfx.'"><label
for="mod_login_remember'.$class_sfx.'">'._UE_REMEMBER_ME."</label></span
>";

                    if ($horizontal) echo " \n"; else echo "<br />\n";

                break;

                default:
```

```
		break;
	}
	$pluginDisplays = array();
	if ( $cb_plugins ) {
		include_once( $absolute_path .
"/administrator/components/com_comprofiler/plugin.class.php");
		global $_PLUGINS;
		$_PLUGINS->loadPluginGroup('user');
		$pluginsResults = $_PLUGINS->trigger(
'onAfterLoginForm', array( $name_lenght, $pass_lenght, $horizontal, $class_sfx,
&$params ) );
		if ( count( $pluginsResults ) > 0 ) {
			foreach ( $pluginsResults as $pR ) {
				if ( is_array( $pR ) ) {
					foreach ($pR as $pK => $pV ) {
$pluginDisplays[$pK][] = $pV;
					}
				} elseif ( $pR != '' ) {
$pluginDisplays['beforeButton'][] = $pR;
				}
			}
```

```
                }

                foreach ( $pluginDisplays as $pK => $pV ) {

                        $divHtml                        =       ( $horizontal ?
'<span' : '<div' ) . ' class="mod_login_plugin'.$class_sfx.' mod_login_plugin_' . $pK .
'">';

                        $sldivHtml                      =       ( $horizontal ?
'</span>' : '</div>' );

                        $pluginDisplays[$pK]    =       $divHtml . implode(
$sldivHtml . $divHtml, $pV ) . $sldivHtml;

                }

                        if ( isset( $pluginDisplays['beforeButton'] ) ) {

                        echo $pluginDisplays['beforeButton'];

                }

        }

        // Login button/icon:

        switch ( $show_buttons_icons ) {

                case 2:

                        $buttonStyle    =       ' style="width:25px;height:20px;border-
width:0px;margin:0px;cursor:pointer;vertical-align:top;background-image:url(' .
$urlImgPath . 'login.png);background-position:0 0;background-repeat:no-repeat;"'

                        ' title="' . _UE_BUTTON_LOGIN . '"';

                        $buttonValue  =         '';
```

```
                    break;
                case 1:
                    $buttonStyle = ' style="min-height:20px;padding-left:30px;cursor:pointer;background-image:url(' . $urlImgPath . 'login.png);background-position:0 0;background-repeat:no-repeat;width:auto;"';
                    $buttonValue = _UE_BUTTON_LOGIN;
                    break;
                case 0:
                default:
                    $buttonStyle = '';
                    $buttonValue = _UE_BUTTON_LOGIN;
                    break;
            }
            echo '<input type="submit" name="Submit" class="button'.$class_sfx.'" value="' . $buttonValue . '" ' . $buttonStyle . ' />';
            if ($horizontal || ( $show_remind_register_icons == 2 ) ) {
                echo "  \n";
            } else {
                echo "</td></tr>\n<tr><td>";
            }
            if ( isset( $pluginDisplays['afterButton'] ) ) {
```

```
            echo $pluginDisplays['afterButton'];
        }
        if ($show_lostpass) {
            $loginPost = $_CB_framework->viewUrl( 'lostpassword' );
            if ( $https_post /* && ! $isHttps */ ) {
                if ( ( substr($loginPost, 0, 5) != "http:" ) && ( substr($loginPost, 0, 6) != "https:" ) ) {
                    $loginPost = $cblogin_live_site."/".$loginPost;
                }
                $loginPost = str_replace("http://","https://",$loginPost);
            }
            echo '<a href="'.$loginPost.'" class="mod_login'.$class_sfx.'">';
            if ( $show_remind_register_icons >= 1 ) {
                echo '<img src="' . $urlImgPath . 'forgot.png" alt="' . _UE_USERNAME_PASSWORD_REMINDER . '" title="' . _UE_USERNAME_PASSWORD_REMINDER . '" width="25px" height="20px" style="border-width:0px;cursor:pointer;" /> ';
            }
            if ( $show_remind_register_icons <= 1 ) {
                if ($compact) {
                    echo _UE_LOST_USERNAME_PASSWORD;
```

```
                        } else {
                                echo ( ( checkJversion() == -1 ) ?
_UE_USERNAME_PASSWORD_REMINDER:
_UE_LOST_USERNAME_PASSWORD );
                        }
                }
                echo '</a>';
                if ( $show_remind_register_icons == 2 ) {
                        echo " \n";
                } elseif ($horizontal) {
                        if ($compact) {
                                echo " |";
                        } else {
                                echo " \n";
                        }
                } else echo "</td></tr>\n";
        }
        // CB config may override the system configuration setting
        $registration_enabled =         $_CB_framework->getCfg(
'allowUserRegistration' );
        if ( ! $registration_enabled ) {
```

```
                if ( isset($ueConfig['reg_admin_allowcbregistration']) &&
$ueConfig['reg_admin_allowcbregistration'] == '1' ) {
                        $registration_enabled = true;
                }
        }
        if ($registration_enabled && $show_newaccount) {
                if ($horizontal || ( $show_remind_register_icons == 2 ) ) {
                        echo ' <span id="mod_login_noaccount'.$id_sfx.'">';
                } else {
                        echo "<tr><td>";
                }
                if ( ( ! $compact ) && ( $show_remind_register_icons == 0 ) ) {
                        echo _UE_NO_ACCOUNT . " ";
                }
                $loginPost                      =       $_CB_framework->viewUrl(
'registers' );
                if ( $https_post /* && ! $isHttps */ ) {
if ( ( substr($loginPost, 0, 5) != "http:" ) && ( substr($loginPost, 0, 6) != "https:" ) ) {
                                $loginPost = $cblogin_live_site."/".$loginPost;
                        }
                        $loginPost = str_replace("http://","https://",$loginPost);
```

```
                }
                echo '<a href="'.$loginPost.'" class="mod_login'.$class_sfx.'">';
                if ( $show_remind_register_icons >= 1 ) {
                        echo '<img src="' . $urlImgPath . 'register.png" alt="' .
_UE_REGISTER . '" title="' . _UE_REGISTER . '" width="25px" height="20px"
style="border-width:0px;cursor:pointer;" /> ';
                }
                if ( $show_remind_register_icons <= 1 ) {
                        echo ( ( ( checkJversion() == -1 ) && ! $compact ) ?
_UE_CREATE_ACCOUNT : _UE_REGISTER );
                }
                echo '</a>';
                if ($horizontal || ( $show_remind_register_icons == 2 ) ) {
                        echo "</span>\n";
                }
                if ( ! $horizontal ) {
                        echo "</td></tr>\n";
                }
        }
        if (!$horizontal) {
                echo "</table>";
```

```
		}
		echo "</form>";
		if ( isset( $pluginDisplays['almostEnd'] ) ) {
			echo $pluginDisplays['almostEnd'];
		}
		if ( $posttext ) {
			if ( defined( $posttext ) ) {
				echo constant( $posttext );
			} else {
				echo $posttext;
			}
			echo "\n";
		}
}	?>
```

C:\wamp\www\mashup\modules\mod_footer\mod_footer.php

```
<?php
 /**
  * @version		$Id: mod_footer.php 10381 2008-06-01 03:35:53Z pasamio $
  * @package		Joomla
  * @copyright		Copyright (C) 2005 - 2008 Open Source Matters. All rights
```

```
 reserved.
 * @license        GNU/GPL, see LICENSE.php
 * Joomla! is free software. This version may have been modified pursuant
 * to the GNU General Public License, and as distributed it includes or
 * is derivative of works licensed under the GNU General Public License or
 * other free or open source software licenses.
 * See COPYRIGHT.php for copyright notices and details.
 */
 // no direct access
 defined('_JEXEC') or die('Restricted access');
 global $mainframe;
 $date =& JFactory::getDate();
 $cur_year      = $date->toFormat('%Y');
 $csite_name  = $mainframe->getCfg('sitename');
if (JString::strpos(JText :: _('FOOTER_LINE1'), '%date%')) {
        $line1 = ereg_replace('%date%', $cur_year, JText :: _('FOOTER_LINE1'));
} else {
        $line1 = JText :: _('FOOTER_LINE1');
}
if (JString::strpos($line1, '%sitename%')) {
        $lineone = ereg_replace('%sitename%', $csite_name, $line1);
} else {
        $lineone = $line1;
```

}

require(JModuleHelper::getLayoutPath('mod_footer'));

?>

C:\wamp\www\mashup\modules\mod_latestnews\mod_latestnews.php

```
<?php
/**
* @version $Id: mod_latestnews.php 10381 2008-06-01 03:35:53Z pasamio
$
* @package  Joomla
* @copyright Copyright (C) 2005 - 2008 Open Source Matters. All rights
reserved.
* @license    GNU/GPL, see LICENSE.php
* Joomla! is free software. This version may have been modified pursuant
* to the GNU General Public License, and as distributed it includes or
* is derivative of works licensed under the GNU General Public License or
* other free or open source software licenses.
* See COPYRIGHT.php for copyright notices and details.
*/
// no direct access
defined('_JEXEC') or die('Restricted access');
// Include the syndicate functions only once
require_once (dirname(__FILE__).DS.'helper.php');
$list = modLatestNewsHelper::getList($params);
        require(JModuleHelper::getLayoutPath('mod_latestnews'));
```

C:\wamp\www\mashup\modules\mod_comprofileronline\mod_comprofileronline.php

```
<?php
/**
* Users Online Module 1.2
* $Id: mod_comprofileronline.php 1113 2010-06-22 17:37:40Z beat $
* @version 1.2
* @package Community Builder 1.2
* @Copyright (C) 2004-2010 Beat and 2000 - 2003 Miro International Pty Ltd
* @ All rights reserved
* @ Mambo Open Source is Free Software
* @ Released under GNU/GPL License : http://www.gnu.org/copyleft/gpl.html
**/
if ( ! ( defined( '_VALID_CB' ) || defined( '_JEXEC' ) || defined( '_VALID_MOS' ) ) ) { die( 'Direct Access to this location is not allowed.' ); }
/**
 * CB framework
 * @global CBframework $_CB_framework
 */
global $_CB_framework, $_CB_database, $ueConfig, $mainframe;
if ( defined( 'JPATH_ADMINISTRATOR' ) ) {
        if ( ! file_exists( JPATH_ADMINISTRATOR .
                return;
        }
        include_once( JPATH_ADMINISTRATOR .
'/components/com_comprofiler/plugin.foundation.php' );
```

```
'/components/com_comprofiler/plugin.foundation.php' ) ) {
                echo 'CB not installed';
return;
        }
include_once( JPATH_ADMINISTRATOR .
'/components/com_comprofiler/plugin.foundation.php' );
} else {
        if ( ! file_exists( $mainframe->getCfg( 'absolute_path' ) .
'/administrator/components/com_comprofiler/plugin.foundation.php' ) ) {
                echo 'CB not installed';
                return;
        }
        include_once( $mainframe->getCfg( 'absolute_path' ) .
'/administrator/components/com_comprofiler/plugin.foundation.php' );
}
cbimport( 'cb.database' );
cbimport( 'language.front' );

function getNameFormatOnline($name,$uname,$format) {
        if ( $format != 3 ) {
```

```
		$name =		str_replace(
array("&amp;","&quot;","&#039;","&lt;","&gt;"), array("&","\"","'","<",">"), $name
);
	}
	SWITCH ($format) {
		CASE 1 :
			$returnName = $name;
		break;
		CASE 2 :
			$returnName = $name." (".$uname.")";
		break;
		CASE 3 :
			$returnName = $uname;
		break;
		CASE 4 :
			$returnName = $uname." (".$name.")";
		break;
	}
	return $returnName;
}
// $params is defined by include: ignore this warning:
```

```
if (is_callable(array($params,"get"))) {        // Mambo 4.5.0 compatibility
        $class_sfx      =       $params->get( 'moduleclass_sfx');
        $pretext        =       $params->get( 'pretext', "" );
        $posttext       =       $params->get( 'posttext', "" );
} else {
        $class_sfx      =       '';
        $pretext        =       '';
        $posttext       =       '';
}
$query  =       "SELECT DISTINCT a.username, a.userid, u.name"
."\n FROM #__session AS a, #__users AS u"
."\n WHERE (a.userid = u.id) AND (a.guest = 0) AND "
.       ( ( checkJversion() >= 1 ) ? "(a.client_id = 0)" : "(NOT ( a.usertype is NULL OR a.usertype = ''))" )
."\n ORDER BY " . ( ( $ueConfig['name_format'] > 2 ) ? "a.username" : "u.name" ) . " ASC";
$_CB_database->setQuery($query);
$rows   =       $_CB_database->loadObjectList();

$result  =       '';
if ( count( $rows ) > 0) {
```

```
$result.="<ul class='mod_login".$class_sfx."'>\n";  // style='list-style-type:none;
margin:0px; padding:0px; font-weight:bold;'

        foreach($rows as $row) {

        $result.="<li><a href='" . $_CB_framework->userProfileUrl( (int) $row-
>userid ).        "'
class='mod_login".$class_sfx."'>".htmlspecialchars(getNameFormatOnline($row-
>name,$row->username,$ueConfig['name_format'])."</a></li>\n";

        }

        $result  .=       "</ul>\n";

        if ( $pretext != '' ) {

                $result= $pretext . "<br />\n" . $result;

        }

        $result .=$posttext;

} else {

        $result           .=       _UE_NONE;

}

echo $result;

?>

$server->wsdl->schemaTargetNamespace = 'urn:server';

// configuracion del  metodo

$server->register('usuarios',
```

```
        array('usuario' => 'xsd:string'),
        array('return' => 'xsd:string'),
        'urn:server',
        'urn:server#usuarios'
);



//metodos de consultas
function usuarios($usuario){
        require("conexion.php");
          $data="0";
                $rs = mysql_query("SELECT * FROM jos_users  where
username='".$usuario."'", $con);
                 if(mysql_num_rows($rs)>0){
                        $data="1";
                 }
               return $data;
}
```

```
// Use the request to (try to) invoke the service

$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ?
$HTTP_RAW_POST_DATA : '';

$server->service($HTTP_RAW_POST_DATA);

?>

<?php

require_once('DataBasePOO.php');

$db = new DataBasePOO();

echo "111";

$data="";

$datos=$db->Listado("SELECT username FROM jos_users");


        for ($i=0;$i<count($datos); $i++){

                                $row=$datos[$i];

                            //          print_r($row);

                                        $data=$data.$row[0];

                                                    }

                    echo $data;


?>
```

<?php

/*

$Id: nusoapmime.php,v 1.13 2010/04/26 20:15:08 snichol Exp $

NuSOAP - Web Services Toolkit for PHP

Copyright (c) 2002 NuSphere Corporation

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

The NuSOAP project home is:

http://sourceforge.net/projects/nusoap/

The primary support for NuSOAP is the mailing list: nusoap general@lists.sourceforge.net

If you have any questions or comments, please email:

Dietrich Ayala

dietrich@ganx4.com

http://dietrich.ganx4.com/nusoap

NuSphere Corporation

http://www.nusphere.com

*/

/*require_once('nusoap.php');*/

/* PEAR Mail_MIME library */

require_once('Mail/mimeDecode.php');

require_once('Mail/mimePart.php');

```
/**
* nusoap_client_mime client supporting MIME attachments defined at
* http://www.w3.org/TR/SOAP-attachments.  It depends on the PEAR Mail_MIME library.
*
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @author       Thanks to Guillaume and Henning Reich for posting great attachment code to the mail list
* @version  $Id: nusoapmime.php,v 1.13 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class nusoap_client_mime extends nusoap_client {
        /**
         * @var array Each array element in the return is an associative array with keys
         * data, filename, contenttype, cid
         * @access private
         */
        var $requestAttachments = array();
        /**
```

```
     * @var array Each array element in the return is an associative array with
keys
     * data, filename, contenttype, cid
     * @access private
     */
    var $responseAttachments;
    /**
     * @var string
     * @access private
     */
    var $mimeContentType;


    /**
    * adds a MIME attachment to the current request.
    *
    * If the $data parameter contains an empty string, this method will read
    * the contents of the file named by the $filename parameter.
    *
    * If the $cid parameter is false, this method will generate the cid.
    *
```

```
        * @param string $data The data of the attachment

        * @param string $filename The filename of the attachment (default is empty
string)

        * @param string $contenttype The MIME Content-Type of the attachment
(default is application/octet-stream)

        * @param string $cid The content-id (cid) of the attachment (default is false)

        * @return string The content-id (cid) of the attachment

        * @access public

        */

        function addAttachment($data, $filename = '', $contenttype =
'application/octet-stream', $cid = false) {

                if (! $cid) {

                        $cid = md5(uniqid(time()));

                }


                $info['data'] = $data;

                $info['filename'] = $filename;

                $info['contenttype'] = $contenttype;

                $info['cid'] = $cid;


                $this->requestAttachments[] = $info;
```

```
        return $cid;
}

/**
* clears the MIME attachments for the current request.
*
* @access public
*/
function clearAttachments() {
        $this->requestAttachments = array();
}

/**
* gets the MIME attachments from the current response.
*
* Each array element in the return is an associative array with keys
* data, filename, contenttype, cid.  These keys correspond to the parameters
* for addAttachment.
*
* @return array The attachments.
```

```
* @access public
*/
function getAttachments() {
        return $this->responseAttachments;
}

/**
* gets the HTTP body for the current request.
*
* @param string $soapmsg The SOAP payload
* @return string The HTTP body, which includes the SOAP payload
* @access private
*/
function getHTTPBody($soapmsg) {
        if (count($this->requestAttachments) > 0) {
                $params['content_type'] = 'multipart/related; type="text/xml"';
                $mimeMessage = new Mail_mimePart('', $params);
                unset($params);

                $params['content_type'] = 'text/xml';
```

```
                $params['encoding']     = '8bit';
                $params['charset']      = $this->soap_defencoding;
                $mimeMessage->addSubpart($soapmsg, $params);

                foreach ($this->requestAttachments as $att) {
                        unset($params);

                        $params['content_type'] = $att['contenttype'];
                        $params['encoding']     = 'base64';
                        $params['disposition']  = 'attachment';
                        $params['dfilename']    = $att['filename'];
                        $params['cid']          = $att['cid'];

                        if ($att['data'] == '' && $att['filename'] <> '') {
                                if ($fd = fopen($att['filename'], 'rb')) {
                                        $data = fread($fd,
filesize($att['filename']));
                                        fclose($fd);
                                } else {
                                        $data = '';
```

```
                }
                $mimeMessage->addSubpart($data, $params);
            } else {
                $mimeMessage->addSubpart($att['data'],
$params);
            }
        }

        $output = $mimeMessage->encode();
        $mimeHeaders = $output['headers'];

        foreach ($mimeHeaders as $k => $v) {
            $this->debug("MIME header $k: $v");
            if (strtolower($k) == 'content-type') {
                // PHP header() seems to strip leading
whitespace starting
                // the second line, so force everything to one line
                $this->mimeContentType = str_replace("\r\n", "
", $v);
            }
        }
```

```
                return $output['body'];
        }

        return parent::getHTTPBody($soapmsg);
}

/**
* gets the HTTP content type for the current request.
*
* Note: getHTTPBody must be called before this.
*
* @return string the HTTP content type for the current request.
* @access private
*/
function getHTTPContentType() {
        if (count($this->requestAttachments) > 0) {
                return $this->mimeContentType;
        }
        return parent::getHTTPContentType();
}
```

```
        /**
* gets the HTTP content type charset for the current request.
* returns false for non-text content types.
*
* Note: getHTTPBody must be called before this.
*
* @return string the HTTP content type charset for the current request.
* @access private
*/
function getHTTPContentTypeCharset() {
        if (count($this->requestAttachments) > 0) {
                return false;
        }
        return parent::getHTTPContentTypeCharset();
}

/**
* processes SOAP message returned from server
*
* @param       array   $headers        The HTTP headers
```

```
        * @param      string   $data              unprocessed response data from server
        * @return     mixed  value of the message, decoded into a PHP type
        * @access   private
        */
    function parseResponse($headers, $data) {
                $this->debug('Entering parseResponse() for payload of length ' .
strlen($data) . ' and type of ' . $headers['content-type']);
                $this->responseAttachments = array();
                if (strstr($headers['content-type'], 'multipart/related')) {
                        $this->debug('Decode multipart/related');
                        $input = '';
                        foreach ($headers as $k => $v) {
                                $input .= "$k: $v\r\n";
                        }
                        $params['input'] = $input . "\r\n" . $data;
                        $params['include_bodies'] = true;
                        $params['decode_bodies'] = true;
                        $params['decode_headers'] = true;

                        $structure = Mail_mimeDecode::decode($params);
```

```
                foreach ($structure->parts as $part) {
                        if (!isset($part->disposition) && (strstr($part->headers['content-type'], 'text/xml'))) {
                                $this->debug('Have root part of type ' . $part->headers['content-type']);
                                $root = $part->body;
                                $return = parent::parseResponse($part->headers, $part->body);
                        } else {
                                $this->debug('Have an attachment of type ' . $part->headers['content-type']);
                                $info['data'] = $part->body;
                                $info['filename'] = isset($part->d_parameters['filename']) ? $part->d_parameters['filename'] : '';
                                $info['contenttype'] = $part->headers['content-type'];
                                $info['cid'] = $part->headers['content-id'];
                                $this->responseAttachments[] = $info;
                        }
                }

                if (isset($return)) {
```

```
                    $this->responseData = $root;
                    return $return;
                }

                $this->setError('No root part found in multipart/related content');
                return '';
            }
            $this->debug('Not multipart/related');
            return parent::parseResponse($headers, $data);
        }
}

/*
 *      For backwards compatiblity, define soapclientmime unless the PHP SOAP extension is loaded.
 */
if (!extension_loaded('soap')) {
        class soapclientmime extends nusoap_client_mime {
        }
}
```

```
/**
* nusoap_server_mime server supporting MIME attachments defined at
* http://www.w3.org/TR/SOAP-attachments.  It depends on the PEAR Mail_MIME library.
*
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @author       Thanks to Guillaume and Henning Reich for posting great attachment code to the mail list
* @version  $Id: nusoapmime.php,v 1.13 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class nusoap_server_mime extends nusoap_server {
        /**
         * @var array Each array element in the return is an associative array with keys
         * data, filename, contenttype, cid
         * @access private
         */
        var $requestAttachments = array();
        /**
```

```
     * @var array Each array element in the return is an associative array with
keys
     * data, filename, contenttype, cid
     * @access private
     */
    var $responseAttachments;
    /**
     * @var string
     * @access private
     */
    var $mimeContentType;


    /**
    * adds a MIME attachment to the current response.
    *
    * If the $data parameter contains an empty string, this method will read
    * the contents of the file named by the $filename parameter.
    *
    * If the $cid parameter is false, this method will generate the cid.
    *
```

```
        * @param string $data The data of the attachment

        * @param string $filename The filename of the attachment (default is empty
string)

        * @param string $contenttype The MIME Content-Type of the attachment
(default is application/octet-stream)

        * @param string $cid The content-id (cid) of the attachment (default is false)

        * @return string The content-id (cid) of the attachment

        * @access public

        */

        function addAttachment($data, $filename = '', $contenttype =
'application/octet-stream', $cid = false) {

                if (! $cid) {

                        $cid = md5(uniqid(time()));

                }

                $info['data'] = $data;

                $info['filename'] = $filename;

                $info['contenttype'] = $contenttype;

                $info['cid'] = $cid;

                $this->responseAttachments[] = $info;

                return $cid;

        }
```

```
/**
* clears the MIME attachments for the current response.
*
* @access public
*/
function clearAttachments() {
        $this->responseAttachments = array();
}


/**
* gets the MIME attachments from the current request.
*
* Each array element in the return is an associative array with keys
* data, filename, contenttype, cid.  These keys correspond to the parameters
* for addAttachment.
*
* @return array The attachments.
* @access public
*/
```

```
function getAttachments() {
        return $this->requestAttachments;
}

/**
* gets the HTTP body for the current response.
*
* @param string $soapmsg The SOAP payload
* @return string The HTTP body, which includes the SOAP payload
* @access private
*/
function getHTTPBody($soapmsg) {
        if (count($this->responseAttachments) > 0) {
                $params['content_type'] = 'multipart/related; type="text/xml"';
                $mimeMessage = new Mail_mimePart('', $params);
                unset($params);

                $params['content_type'] = 'text/xml';
                $params['encoding']     = '8bit';
                $params['charset']      = $this->soap_defencoding;
```

```
$mimeMessage->addSubpart($soapmsg, $params);

foreach ($this->responseAttachments as $att) {
    unset($params);

    $params['content_type'] = $att['contenttype'];
    $params['encoding']     = 'base64';
    $params['disposition']  = 'attachment';
    $params['dfilename']    = $att['filename'];
    $params['cid']          = $att['cid'];

    if ($att['data'] == '' && $att['filename'] <> '') {
        if ($fd = fopen($att['filename'], 'rb')) {
            $data = fread($fd,
filesize($att['filename']));
            fclose($fd);
        } else {
            $data = '';
        }
        $mimeMessage->addSubpart($data, $params);
```

```
                } else {
                        $mimeMessage->addSubpart($att['data'],
$params);
                }
        }

        $output = $mimeMessage->encode();
        $mimeHeaders = $output['headers'];

        foreach ($mimeHeaders as $k => $v) {
                $this->debug("MIME header $k: $v");
                if (strtolower($k) == 'content-type') {
                        // PHP header() seems to strip leading
whitespace starting
                        // the second line, so force everything to one line
                        $this->mimeContentType = str_replace("\r\n", "
", $v);
                }
        }

        return $output['body'];
```

```
        }

        return parent::getHTTPBody($soapmsg);
}

/**
* gets the HTTP content type for the current response.
*
* Note: getHTTPBody must be called before this.
*
* @return string the HTTP content type for the current response.
* @access private
*/
function getHTTPContentType() {
        if (count($this->responseAttachments) > 0) {
                return $this->mimeContentType;
        }
        return parent::getHTTPContentType();
}
```

```
/**
* gets the HTTP content type charset for the current response.
* returns false for non-text content types.
*
* Note: getHTTPBody must be called before this.
*
* @return string the HTTP content type charset for the current response.
* @access private
*/
function getHTTPContentTypeCharset() {
        if (count($this->responseAttachments) > 0) {
                return false;
        }
        return parent::getHTTPContentTypeCharset();
}

/**
* processes SOAP message received from client
*
* @param       array    $headers         The HTTP headers
```

```
        * @param      string   $data              unprocessed request data from client
        * @return     mixed  value of the message, decoded into a PHP type
        * @access   private
        */
    function parseRequest($headers, $data) {
                $this->debug('Entering parseRequest() for payload of length ' .
strlen($data) . ' and type of ' . $headers['content-type']);
                $this->requestAttachments = array();
                if (strstr($headers['content-type'], 'multipart/related')) {
                        $this->debug('Decode multipart/related');
                        $input = '';
                        foreach ($headers as $k => $v) {
                                $input .= "$k: $v\r\n";
                        }
                        $params['input'] = $input . "\r\n" . $data;
                        $params['include_bodies'] = true;
                        $params['decode_bodies'] = true;
                        $params['decode_headers'] = true;

                        $structure = Mail_mimeDecode::decode($params);
```

```
                foreach ($structure->parts as $part) {
                        if (!isset($part->disposition) && (strstr($part->headers['content-type'], 'text/xml'))) {
                                $this->debug('Have root part of type ' . $part->headers['content-type']);
                                $return = parent::parseRequest($part->headers, $part->body);
                        } else {
                                $this->debug('Have an attachment of type ' . $part->headers['content-type']);
                                $info['data'] = $part->body;
                                $info['filename'] = isset($part->d_parameters['filename']) ? $part->d_parameters['filename'] : '';
                                $info['contenttype'] = $part->headers['content-type'];
                                $info['cid'] = $part->headers['content-id'];
                                $this->requestAttachments[] = $info;
                        }
                }
                        if (isset($return)) {
                        return $return;
                }
```

```
                        $this->setError('No root part found in multipart/related
content');
                        return;
                }
                $this->debug('Not multipart/related');
                return parent::parseRequest($headers, $data);
        }
}

/*
 *      For backwards compatiblity
 */
class nusoapservermime extends nusoap_server_mime {
}
```

?>

<?php

/*

$Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $

NuSOAP - Web Services Toolkit for PHP

Copyright (c) 2002 NuSphere Corporation

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

The NuSOAP project home is:

http://sourceforge.net/projects/nusoap/

The primary support for NuSOAP is the Help forum on the project home page.

If you have any questions or comments, please email:

Dietrich Ayala

dietrich@ganx4.com

http://dietrich.ganx4.com/nusoap

NuSphere Corporation

http://www.nusphere.com

*/

/*

 *	Some of the standards implmented in whole or part by NuSOAP:

 *

```
 *      SOAP 1.1 (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)
 *      WSDL 1.1 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315)
 *      SOAP Messages With Attachments (http://www.w3.org/TR/SOAP-
attachments)
 *      XML 1.0 (http://www.w3.org/TR/2006/REC-xml-20060816/)
 *      Namespaces in XML 1.0 (http://www.w3.org/TR/2006/REC-xml-names-
20060816/)
 *      XML Schema 1.0 (http://www.w3.org/TR/xmlschema-0/)
 *      RFC 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format
of Internet Message Bodies
 *      RFC 2068 Hypertext Transfer Protocol -- HTTP/1.1
 *      RFC 2617 HTTP Authentication: Basic and Digest Access Authentication
 */

/* load classes

// necessary classes
require_once('class.soapclient.php');
require_once('class.soap_val.php');
require_once('class.soap_parser.php');
require_once('class.soap_fault.php');
```

```
// transport classes
require_once('class.soap_transport_http.php');

// optional add-on classes
require_once('class.xmlschema.php');
require_once('class.wsdl.php');

// server class
require_once('class.soap_server.php');*/

// class variable emulation
// cf. http://www.webkreator.com/php/techniques/php-static-class-variables.html
$GLOBALS['_transient']['static']['nusoap_base']['globalDebugLevel'] = 9;

/**
*
* nusoap_base
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
```

```
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class nusoap_base {
        /**
         * Identification for HTTP headers.
         *
         * @var string
         * @access private
         */
        var $title = 'NuSOAP';
        /**
         * Version for HTTP headers.
         *
         * @var string
         * @access private
         */
        var $version = '0.9.5';
        /**
         * CVS revision for HTTP headers.
```

```
        *
        * @var string
        * @access private
        */
       var $revision = '$Revision: 1.123 $';
   /**
    * Current error string (manipulated by getError/setError)
        *
        * @var string
        * @access private
        */
       var $error_str = '';
   /**
    * Current debug string (manipulated by
debug/appendDebug/clearDebug/getDebug/getDebugAsXMLComment)
        *
        * @var string
        * @access private
        */
   var $debug_str = '';
```

```
/**
     * toggles automatic encoding of special characters as entities
     * (should always be true, I think)
     *
     * @var boolean
     * @access private
     */
    var $charencoding = true;
    /**
     * the debug level for this instance
     *
     * @var          integer
     * @access private
     */
    var $debugLevel;

/**
    * set schema version
    *
    * @var      string
```

```
        * @access   public
        */
        var $XMLSchemaVersion = 'http://www.w3.org/2001/XMLSchema';

/**
        * charset encoding for outgoing messages
        *
        * @var      string
        * @access   public
        */
var $soap_defencoding = 'ISO-8859-1';
        //var $soap_defencoding = 'UTF-8';

        /**
        * namespaces in an array of prefix => uri
        *
        * this is "seeded" by a set of constants, but it may be altered by code
        *
        * @var      array
        * @access   public
```

```
*/
var $namespaces = array(
        'SOAP-ENV' => 'http://schemas.xmlsoap.org/soap/envelope/',
        'xsd' => 'http://www.w3.org/2001/XMLSchema',
        'xsi' => 'http://www.w3.org/2001/XMLSchema-instance',
        'SOAP-ENC' => 'http://schemas.xmlsoap.org/soap/encoding/'
        );

/**
* namespaces used in the current context, e.g. during serialization
*
* @var      array
* @access   private
*/
var $usedNamespaces = array();

/**
* XML Schema types in an array of uri => (array of xml type => php type)
* is this legacy yet?
```

```
        * no, this is used by the nusoap_xmlschema class to verify type => namespace
mappings.
        * @var      array
        * @access   public
        */
        var $typemap = array(
        'http://www.w3.org/2001/XMLSchema' => array(

        'string'=>'string','boolean'=>'boolean','float'=>'double','double'=>'double','deci
mal'=>'double',

        'duration'=>'','dateTime'=>'string','time'=>'string','date'=>'string','gYearMonth'
=>'',

        'gYear'=>'','gMonthDay'=>'','gDay'=>'','gMonth'=>'','hexBinary'=>'string','base
64Binary'=>'string',
                // abstract "any" types
                'anyType'=>'string','anySimpleType'=>'string',
                // derived datatypes

        'normalizedString'=>'string','token'=>'string','language'=>'','NMTOKEN'=>'','
NMTOKENS'=>'','Name'=>'','NCName'=>'','ID'=>'',
```

'IDREF'=>'','IDREFS'=>'','ENTITY'=>'','ENTITIES'=>'','integer'=>'integer','nonPositiveInteger'=>'integer',

'negativeInteger'=>'integer','long'=>'integer','int'=>'integer','short'=>'integer','byte'=>'integer','nonNegativeInteger'=>'integer',

'unsignedLong'=>'','unsignedInt'=>'','unsignedShort'=>'','unsignedByte'=>'','positiveInteger'=>''),

'http://www.w3.org/2000/10/XMLSchema' => array(

'i4'=>'','int'=>'integer','boolean'=>'boolean','string'=>'string','double'=>'double',

'float'=>'double','dateTime'=>'string',

'timeInstant'=>'string','base64Binary'=>'string','base64'=>'string','urtype'=>'array'),

'http://www.w3.org/1999/XMLSchema' => array(

'i4'=>'','int'=>'integer','boolean'=>'boolean','string'=>'string','double'=>'double',

'float'=>'double','dateTime'=>'string',

'timeInstant'=>'string','base64Binary'=>'string','base64'=>'string','urtype'=>'array'),

'http://soapinterop.org/xsd' => array('SOAPStruct'=>'struct'),

'http://schemas.xmlsoap.org/soap/encoding/' =>
array('base64'=>'string','array'=>'array','Array'=>'array'),

```
'http://xml.apache.org/xml-soap' => array('Map')
    );

    /**
    * XML entities to convert
    *
    * @var      array
    * @access   public
    * @deprecated
    * @seeexpandEntities
    */
    var $xmlEntities = array('quot' => '"','amp' => '&',
            'lt' => '<','gt' => '>','apos' => "'");

    /**
    * constructor
    *
    * @access       public
    */
    function nusoap_base() {
```

```
        $this->debugLevel =
$GLOBALS['_transient']['static']['nusoap_base']['globalDebugLevel'];
    }

    /**
    * gets the global debug level, which applies to future instances
    *
    * @return   integer Debug level 0-9, where 0 turns off
    * @access   public
    */
    function getGlobalDebugLevel() {
        return
$GLOBALS['_transient']['static']['nusoap_base']['globalDebugLevel'];
    }

    /**
    * sets the global debug level, which applies to future instances
    *
    * @param    int     $level  Debug level 0-9, where 0 turns off
    * @access   public
    */
```

```
        function setGlobalDebugLevel($level) {
                $GLOBALS['_transient']['static']['nusoap_base']['globalDebugLevel']
= $level;
        }

        /**
        * gets the debug level for this instance
        *
        * @return       int       Debug level 0-9, where 0 turns off
        * @access       public
        */
        function getDebugLevel() {
                return $this->debugLevel;
        }

        /**
        * sets the debug level for this instance
        *
        * @param        int       $level  Debug level 0-9, where 0 turns off
        * @access       public
```

```
        */
        function setDebugLevel($level) {
                $this->debugLevel = $level;
        }

        /**
        * adds debug data to the instance debug string with formatting
        *
        * @param    string $string debug data
        * @access   private
        */
        function debug($string){
                if ($this->debugLevel > 0) {
                        $this->appendDebug($this->getmicrotime().'
'.get_class($this).": $string\n");
                }
        }

        /**
        * adds debug data to the instance debug string without formatting
```

```
*
* @param    string $string debug data
* @access   public
*/
function appendDebug($string){
        if ($this->debugLevel > 0) {
                // it would be nice to use a memory stream here to use
                // memory more efficiently
                $this->debug_str .= $string;
        }
}

/**
* clears the current debug data for this instance
*
* @access   public
*/
function clearDebug() {
        // it would be nice to use a memory stream here to use
        // memory more efficiently
```

```
        $this->debug_str = '';
}

/**
* gets the current debug data for this instance
*
* @return   debug data
* @access   public
*/
function &getDebug() {
        // it would be nice to use a memory stream here to use
        // memory more efficiently
        return $this->debug_str;
}

/**
* gets the current debug data for this instance as an XML comment
* this may change the contents of the debug data
*
* @return   debug data as an XML comment
```

```
* @access   public
*/
function &getDebugAsXMLComment() {
        // it would be nice to use a memory stream here to use
        // memory more efficiently
        while (strpos($this->debug_str, '--')) {
                $this->debug_str = str_replace('--', '- -', $this->debug_str);
        }
        $ret = "<!--\n" . $this->debug_str . "\n-->";
return $ret;
}

/**
* expands entities, e.g. changes '<' to '&lt;'.
*
* @param       string   $val     The string in which to expand entities.
* @access      private
*/
function expandEntities($val) {
        if ($this->charencoding) {
```

```
            $val = str_replace('&', '&amp;', $val);
            $val = str_replace("'", '&apos;', $val);
            $val = str_replace('"', '&quot;', $val);
            $val = str_replace('<', '&lt;', $val);
            $val = str_replace('>', '&gt;', $val);
        }
        return $val;
    }

    /**
    * returns error string if present
    *
    * @return   mixed error string or false
    * @access   public
    */
    function getError(){
            if($this->error_str != ''){
                    return $this->error_str;
            }
            return false;
```

```
}

/**
* sets error string
*
* @return   boolean $string error string
* @access   private
*/
function setError($str){
        $this->error_str = $str;
}

/**
* detect if array is a simple array or a struct (associative array)
*
* @param      mixed  $val     The PHP array
* @return     string   (arraySimple|arrayStruct)
* @access     private
*/
function isArraySimpleOrStruct($val) {
```

```
        $keyList = array_keys($val);
                foreach ($keyList as $keyListValue) {
                        if (!is_int($keyListValue)) {
                                return 'arrayStruct';
                        }
                }
                return 'arraySimple';
        }

        /**
        * serializes PHP values in accordance w/ section 5. Type information is
        * not serialized if $use == 'literal'.
        *
        * @param      mixed  $val     The value to serialize
        * @param      string   $name The name (local part) of the XML element
        * @param      string   $type   The XML schema type (local part) for the
element
        * @param      string   $name_ns       The namespace for the name of the XML
element
        * @param      string   $type_ns       The namespace for the type of the
element
```

```
        * @param      array    $attributes      The attributes to serialize as
name=>value pairs
        * @param      string   $use     The WSDL "use" (encoded|literal)
        * @param      boolean          $soapval         Whether this is called from
soapval.
        * @return     string   The serialized element, possibly with child elements
    * @access   public
        */
        function
serialize_val($val,$name=false,$type=false,$name_ns=false,$type_ns=false,$attribut
es=false,$use='encoded',$soapval=false) {
                $this->debug("in serialize_val: name=$name, type=$type,
name_ns=$name_ns, type_ns=$type_ns, use=$use, soapval=$soapval");
                $this->appendDebug('value=' . $this->varDump($val));
                $this->appendDebug('attributes=' . $this->varDump($attributes));

        if (is_object($val) && get_class($val) == 'soapval' && (! $soapval)) {
                $this->debug("serialize_val: serialize soapval");
        $xml = $val->serialize($use);
                        $this->appendDebug($val->getDebug());
                        $val->clearDebug();
                        $this->debug("serialize_val of soapval returning $xml");
```

```
                return $xml;
}
        // force valid name if necessary
        if (is_numeric($name)) {
                $name = '__numeric_' . $name;
        } elseif (! $name) {
                $name = 'noname';
        }
        // if name has ns, add ns prefix to name
        $xmlns = '';
if($name_ns){
                $prefix = 'nu'.rand(1000,9999);
                $name = $prefix.':'.$name;
                $xmlns .= " xmlns:$prefix=\"$name_ns\"";
        }
        // if type is prefixed, create type prefix
        if($type_ns != '' && $type_ns == $this->namespaces['xsd']){
                // need to fix this. shouldn't default to xsd if no ns specified
           // w/o checking against typemap
                $type_prefix = 'xsd';
```

```
        } elseif($type_ns){
                $type_prefix = 'ns'.rand(1000,9999);
                $xmlns .= " xmlns:$type_prefix=\"$type_ns\"";
        }
        // serialize attributes if present
        $atts = '';
        if($attributes){
                foreach($attributes as $k => $v){
                        $atts .= " $k=\"".$this->expandEntities($v)."";
                }
        }
        // serialize null value
        if (is_null($val)) {
        $this->debug("serialize_val: serialize null");
                if ($use == 'literal') {
                        // TODO: depends on minOccurs
                        $xml = "<$name$xmlns$atts/>";
                        $this->debug("serialize_val returning $xml");
        return $xml;
} else {
```

```
                                if (isset($type) && isset($type_prefix)) {
                                        $type_str = " xsi:type=\"$type_prefix:$type\"";
                                } else {
                                        $type_str = '';
                                }
                                $xml = "<$name$xmlns$type_str$atts
xsi:nil=\"true\"/>";
                                $this->debug("serialize_val returning $xml");
                        return $xml;
                }
                        }
        // serialize if an xsd built-in primitive type
        if($type != '' && isset($this->typemap[$this->XMLSchemaVersion][$type])){
                        $this->debug("serialize_val: serialize xsd built-in primitive type");
                if (is_bool($val)) {
                        if ($type == 'boolean') {
                                $val = $val ? 'true' : 'false';
                        } elseif (! $val) {
                                $val = 0;
                        }
```

```
                } else if (is_string($val)) {
                        $val = $this->expandEntities($val);
                }
                if ($use == 'literal') {
                        $xml = "<$name$xmlns$atts>$val</$name>";
                        $this->debug("serialize_val returning $xml");
                return $xml;
        } else {
                        $xml = "<$name$xmlns
xsi:type=\"xsd:$type\"$atts>$val</$name>";
                        $this->debug("serialize_val returning $xml");
                return $xml;
        }
    }
                // detect type and serialize
                $xml = '';
                switch(true) {
                        case (is_bool($val) || $type == 'boolean'):
                                $this->debug("serialize_val: serialize boolean");
                if ($type == 'boolean') {
```

```
                $val = $val ? 'true' : 'false';
            } elseif (! $val) {
                $val = 0;
            }
                    if ($use == 'literal') {
                        $xml .= "<$name$xmlns$atts>$val</$name>";
                    } else {
                        $xml .= "<$name$xmlns
xsi:type=\"xsd:boolean\"$atts>$val</$name>";
                    }
                    break;
                case (is_int($val) || is_long($val) || $type == 'int'):
                    $this->debug("serialize_val: serialize int");
                    if ($use == 'literal') {
                        $xml .= "<$name$xmlns$atts>$val</$name>";
                    } else {
                        $xml .= "<$name$xmlns
xsi:type=\"xsd:int\"$atts>$val</$name>";
                    }
                    break;
                case (is_float($val)|| is_double($val) || $type == 'float'):
```

```
                    $this->debug("serialize_val: serialize float");
                    if ($use == 'literal') {
                        $xml .= "<$name$xmlns$atts>$val</$name>";
                    } else {
                        $xml .= "<$name$xmlns
xsi:type=\"xsd:float\"$atts>$val</$name>";
                    }
                    break;
                case (is_string($val) || $type == 'string'):
                    $this->debug("serialize_val: serialize string");
                    $val = $this->expandEntities($val);
                    if ($use == 'literal') {
                        $xml .= "<$name$xmlns$atts>$val</$name>";
                    } else {
                        $xml .= "<$name$xmlns
xsi:type=\"xsd:string\"$atts>$val</$name>";
                    }
                    break;
                case is_object($val):
                    $this->debug("serialize_val: serialize object");
                if (get_class($val) == 'soapval') {
```

```
                $this->debug("serialize_val: serialize soapval object");
            $pXml = $val->serialize($use);
                $this->appendDebug($val->getDebug());
                $val->clearDebug();
        } else {
                if (! $name) {
                    $name = get_class($val);
                    $this->debug("In serialize_val, used class name $name as element name");
                } else {
                    $this->debug("In serialize_val, do not override name $name for element name for class " . get_class($val));
                }
                foreach(get_object_vars($val) as $k => $v){
                    $pXml = isset($pXml) ? $pXml.$this->serialize_val($v,$k,false,false,false,false,$use) : $this->serialize_val($v,$k,false,false,false,false,$use);
                }
            }
            if(isset($type) && isset($type_prefix)){
                $type_str = " xsi:type=\"$type_prefix:$type\"";
```

```
                    } else {
                        $type_str = '';
                    }
                    if ($use == 'literal') {
                        $xml .=
"<$name$xmlns$atts>$pXml</$name>";
                    } else {
                        $xml .=
"<$name$xmlns$type_str$atts>$pXml</$name>";
                    }
                    break;
                break;
                case (is_array($val) || $type):
                    // detect if struct or array
                    $valueType = $this->isArraySimpleOrStruct($val);
            if($valueType=='arraySimple' || preg_match('/^ArrayOf/',$type)){
                        $this->debug("serialize_val: serialize array");
                        $i = 0;
                        if(is_array($val) && count($val)> 0){
                            foreach($val as $v){
                if(is_object($v) && get_class($v) ==  'soapval'){
```

```
                                        $tt_ns = $v->type_ns;
                                        $tt = $v->type;
                                    } elseif (is_array($v)) {
                                        $tt = $this->isArraySimpleOrStruct($v);
                                    } else {
                                        $tt = gettype($v);
                                }
                                    $array_types[$tt] = 1;
                                    // TODO: for literal, the name should be $name
                                    $xml .= $this->serialize_val($v,'item',false,false,false,false,$use);
                                    ++$i;
                                }
                                if(count($array_types) > 1){
                                    $array_typename = 'xsd:anyType';
                                } elseif(isset($tt) && isset($this->typemap[$this->XMLSchemaVersion][$tt])) {
                                    if ($tt == 'integer') {
                                        $tt = 'int';
```

```
                    }
                    $array_typename = 'xsd:'.$tt;
                } elseif(isset($tt) && $tt ==
'arraySimple'){
                    $array_typename = 'SOAP-
ENC:Array';
                } elseif(isset($tt) && $tt ==
'arrayStruct'){
                    $array_typename =
'unnamed_struct_use_soapval';
                } else {
                    // if type is prefixed, create type
prefix
                    if ($tt_ns != '' && $tt_ns == $this-
>namespaces['xsd']){
                        $array_typename = 'xsd:' .
$tt;
                    } elseif ($tt_ns) {
                        $tt_prefix = 'ns' .
rand(1000, 9999);
                        $array_typename =
"$tt_prefix:$tt";
```

```
                                        $xmlns .= "
xmlns:$tt_prefix=\"$tt_ns\"";
                                    } else {
                                        $array_typename = $tt;
                                    }
                                }
                                $array_type = $i;
                                if ($use == 'literal') {
                                    $type_str = '';
                                } else if (isset($type) &&
isset($type_prefix)) {
                                    $type_str = "
xsi:type=\"$type_prefix:$type\"";
                                } else {
                                    $type_str = " xsi:type=\"SOAP-
ENC:Array\" SOAP-ENC:arrayType=\"".$array_typename."[$array_type]\"";
                                }
                            // empty array
                            } else {
                                if ($use == 'literal') {
                                    $type_str = '';
```

```
                } else if (isset($type) &&
isset($type_prefix)) {
                    $type_str = "
xsi:type=\"$type_prefix:$type\"";
                } else {
                    $type_str = " xsi:type=\"SOAP-
ENC:Array\" SOAP-ENC:arrayType=\"xsd:anyType[0]\"";
                }
            }
            // TODO: for array in literal, there is no wrapper
here
            $xml =
"<$name$xmlns$type_str$atts>".$xml."</$name>";
        } else {
            // got a struct
            $this->debug("serialize_val: serialize struct");
            if(isset($type) && isset($type_prefix)){
                $type_str = "
xsi:type=\"$type_prefix:$type\"";
            } else {
                $type_str = '';
            }
```

```
                if ($use == 'literal') {
                        $xml .= "<$name$xmlns$atts>";
                } else {
                        $xml .=
"<$name$xmlns$type_str$atts>";
                }
                foreach($val as $k => $v){
                        // Apache Map
                        if ($type == 'Map' && $type_ns ==
'http://xml.apache.org/xml-soap') {
                                $xml .= '<item>';
                                $xml .= $this-
>serialize_val($k,'key',false,false,false,false,$use);
                                $xml .= $this-
>serialize_val($v,'value',false,false,false,false,$use);
                                $xml .= '</item>';
                        } else {
                                $xml .= $this-
>serialize_val($v,$k,false,false,false,false,$use);
                        }
                }
                $xml .= "</$name>";
```

```
                    }
                    break;
                default:
                    $this->debug("serialize_val: serialize unknown");
                    $xml .= 'not detected, got '.gettype($val).' for '.$val;
                    break;
            }
            $this->debug("serialize_val returning $xml");
            return $xml;
        }

    /**
    * serializes a message
    *
    * @param string $body the XML of the SOAP body
    * @param mixed $headers optional string of XML with SOAP header content, or array of soapval objects for SOAP headers, or associative array
    * @param array $namespaces optional the namespaces used in generating the body and headers
    * @param string $style optional (rpc|document)
    * @param string $use optional (encoded|literal)
```

```
    * @param string $encodingStyle optional (usually
'http://schemas.xmlsoap.org/soap/encoding/' for encoded)
    * @return string the message
    * @access public
    */
    function
serializeEnvelope($body,$headers=false,$namespaces=array(),$style='rpc',$use='enc
oded',$encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'){
    // TODO: add an option to automatically run utf8_encode on $body and $headers
    // if $this->soap_defencoding is UTF-8.  Not doing this automatically allows
    // one to send arbitrary UTF-8 characters, not just characters that map to ISO-8859-
1


        $this->debug("In serializeEnvelope length=" . strlen($body) . " body (max
1000 characters)=" . substr($body, 0, 1000) . " style=$style use=$use
encodingStyle=$encodingStyle");
        $this->debug("headers:");
        $this->appendDebug($this->varDump($headers));
        $this->debug("namespaces:");
        $this->appendDebug($this->varDump($namespaces));


        // serialize namespaces
```

```
    $ns_string = '';
        foreach(array_merge($this->namespaces,$namespaces) as $k => $v){
            $ns_string .= ' xmlns:$k=\"$v\"';
        }
        if($encodingStyle) {
            $ns_string = ' SOAP-
ENV:encodingStyle=\"$encodingStyle\"$ns_string";
        }

        // serialize headers
        if($headers){
            if (is_array($headers)) {
                $xml = '';
                foreach ($headers as $k => $v) {
                    if (is_object($v) && get_class($v) == 'soapval') {
                        $xml .= $this->serialize_val($v, false, false,
false, false, false, $use);
                    } else {
                        $xml .= $this->serialize_val($v, $k, false, false,
false, false, $use);
                    }
```

```
                    }
                    $headers = $xml;
                    $this->debug("In serializeEnvelope, serialized array of headers to $headers");
                }
                $headers = "<SOAP-ENV:Header>".$headers."</SOAP-ENV:Header>";
        }
        // serialize envelope
        return
        '<?xml version="1.0" encoding="'.$this->soap_defencoding .'"?'.">".
        '<SOAP-ENV:Envelope'.$ns_string.">".
        $headers.
        "<SOAP-ENV:Body>".
                $body.
        "</SOAP-ENV:Body>".
        "</SOAP-ENV:Envelope>";
    }

        /**
         * formats a string to be inserted into an HTML stream
```

```
     *
     * @param string $str The string to format
     * @return string The formatted string
     * @access public
     * @deprecated
     */
function formatDump($str){
        $str = htmlspecialchars($str);
        return nl2br($str);
}

    /**
    * contracts (changes namespace to prefix) a qualified name
    *
    * @param    string $qname qname
    * @return       string contracted qname
    * @access   private
    */
    function contractQname($qname){
        // get element namespace
```

```
        //$this->xdebug("Contract $qname");
        if (strrpos($qname, ':')) {
                // get unqualified name
                $name = substr($qname, strrpos($qname, ':') + 1);
                // get ns
                $ns = substr($qname, 0, strrpos($qname, ':'));
                $p = $this->getPrefixFromNamespace($ns);
                if ($p) {
                        return $p . ':' . $name;
                }
                return $qname;
        } else {
                return $qname;
        }
}

/**
* expands (changes prefix to namespace) a qualified name
*
* @param    string $qname qname
```

```
* @return       string expanded qname
* @access   private
*/
function expandQname($qname){
        // get element prefix
        if(strpos($qname,':') && !preg_match('/^http:\/\//',$qname)){
                // get unqualified name
                $name = substr(strstr($qname,':'),1);
                // get ns prefix
                $prefix = substr($qname,0,strpos($qname,':'));
                if(isset($this->namespaces[$prefix])){
                        return $this->namespaces[$prefix].':'.$name;
                } else {
                        return $qname;
                }
        } else {
                return $qname;
        }
}
```

```
/**
* returns the local part of a prefixed string
* returns the original string, if not prefixed
*
* @param string $str The prefixed string
* @return string The local part
* @access public
*/
	function getLocalPart($str){
		if($sstr = strrchr($str,':')){
			// get unqualified name
			return substr( $sstr, 1 );
		} else {
			return $str;
		}
	}

	/**
* returns the prefix part of a prefixed string
* returns false, if not prefixed
```

```
*
* @param string $str The prefixed string
* @return mixed The prefix or false if there is no prefix
* @access public
*/
	function getPrefix($str){
		if($pos = strrpos($str,':')){
			// get prefix
			return substr($str,0,$pos);
		}
		return false;
	}

	/**
* pass it a prefix, it returns a namespace
*
* @param string $prefix The prefix
* @return mixed The namespace, false if no namespace has the specified prefix
* @access public
*/
```

```
    function getNamespaceFromPrefix($prefix){
        if (isset($this->namespaces[$prefix])) {
            return $this->namespaces[$prefix];
        }
        //$this->setError("No namespace registered for prefix '$prefix'");
        return false;
    }

    /**
* returns the prefix for a given namespace (or prefix)
* or false if no prefixes registered for the given namespace
*
* @param string $ns The namespace
* @return mixed The prefix, false if the namespace has no prefixes
* @access public
*/
    function getPrefixFromNamespace($ns) {
        foreach ($this->namespaces as $p => $n) {
            if ($ns == $n || $ns == $p) {
                $this->usedNamespaces[$p] = $n;
```

```
                        return $p;
                    }
                }
                return false;
        }

        /**
* returns the time in ODBC canonical form with microseconds
*
* @return string The time in ODBC canonical form with microseconds
* @access public
*/
        function getmicrotime() {
                if (function_exists('gettimeofday')) {
                        $tod = gettimeofday();
                        $sec = $tod['sec'];
                        $usec = $tod['usec'];
                } else {
                        $sec = time();
                        $usec = 0;
```

```
                }

                return strftime('%Y-%m-%d %H:%M:%S', $sec) . '.' . sprintf('%06d',
$usec);

        }


        /**
         * Returns a string with the output of var_dump
         *
         * @param mixed $data The variable to var_dump
         * @return string The output of var_dump
         * @access public
         */
    function varDump($data) {
                ob_start();
                var_dump($data);
                $ret_val = ob_get_contents();
                ob_end_clean();
                return $ret_val;
        }
```

```
        /**
        * represents the object as a string
        *
        * @return       string
        * @access   public
        */
        function __toString() {
                return $this->varDump($this);
        }
}

// XML Schema Datatype Helper Functions

//xsd:dateTime helpers

/**
* convert unix timestamp to ISO 8601 compliant date string
*
* @param    int $timestamp Unix time stamp
* @param       boolean $utc Whether the time stamp is UTC or local
```

```
* @return       mixed ISO 8601 date string or false
* @access   public
*/
function timestamp_to_iso8601($timestamp,$utc=true){
        $datestr = date('Y-m-d\TH:i:sO',$timestamp);
        $pos = strrpos($datestr, "+");
        if ($pos === FALSE) {
                $pos = strrpos($datestr, "-");
        }
        if ($pos !== FALSE) {
                if (strlen($datestr) == $pos + 5) {
                        $datestr = substr($datestr, 0, $pos + 3) . ':' . substr($datestr, -2);
                }
        }
        if($utc){
                $pattern = '/'.
                '([0-9]{4})-'.   // centuries & years CCYY-
                '([0-9]{2})-'.   // months MM-
                '([0-9]{2})'.    // days DD
                'T'.                        // separator T
```

```
                '([0-9]{2}):'.    // hours hh:
                '([0-9]{2}):'.    // minutes mm:
                '([0-9]{2})(\.[0-9]*)?'. // seconds ss.ss...
                '(Z|[+\-][0-9]{2}:?[0-9]{2})?'. // Z to indicate UTC, -
/+HH:MM:SS.SS... for local tz's
                '/';

                if(preg_match($pattern,$datestr,$regs)){
                        return sprintf('%04d-%02d-
%02dT%02d:%02d:%02dZ',$regs[1],$regs[2],$regs[3],$regs[4],$regs[5],$regs[6]);
                }
                return false;
        } else {
                return $datestr;
        }
}

/**
* convert ISO 8601 compliant date string to unix timestamp
*
* @param    string $datestr ISO 8601 compliant date string
```

```
* @return       mixed Unix timestamp (int) or false
* @access   public
*/
function iso8601_to_timestamp($datestr){
        $pattern = '/'.
        '([0-9]{4})-'.   // centuries & years CCYY-
        '([0-9]{2})-'.   // months MM-
        '([0-9]{2})'.    // days DD
        'T'.                        // separator T
        '([0-9]{2}):'.   // hours hh:
        '([0-9]{2}):'.   // minutes mm:
        '([0-9]{2})(\.[0-9]+)?'. // seconds ss.ss...
        '(Z|[+\-][0-9]{2}:?[0-9]{2})?'. // Z to indicate UTC, -/+HH:MM:SS.SS... for local tz's
        '/';
        if(preg_match($pattern,$datestr,$regs)){
                // not utc
                if($regs[8] != 'Z'){
                        $op = substr($regs[8],0,1);
                        $h = substr($regs[8],1,2);
```

```
                $m = substr($regs[8],strlen($regs[8])-2,2);
                if($op == '-'){
                        $regs[4] = $regs[4] + $h;
                        $regs[5] = $regs[5] + $m;
                } elseif($op == '+'){
                        $regs[4] = $regs[4] - $h;
                        $regs[5] = $regs[5] - $m;
                }
        }
        return gmmktime($regs[4], $regs[5], $regs[6], $regs[2], $regs[3],
$regs[1]);
//      return strtotime("$regs[1]-$regs[2]-$regs[3]
$regs[4]:$regs[5]:$regs[6]Z");
    } else {
        return false;
    }
}

/**
* sleeps some number of microseconds
*
```

```
* @param    string $usec the number of microseconds to sleep
* @access   public
* @deprecated
*/
function usleepWindows($usec)
{
        $start = gettimeofday();

        do
        {
                $stop = gettimeofday();
                $timePassed = 1000000 * ($stop['sec'] - $start['sec'])
                + $stop['usec'] - $start['usec'];
        }
        while ($timePassed < $usec);
}

?><?php
```

```
/**
* Contains information for a SOAP fault.
* Mainly used for returning faults from deployed functions
* in a server instance.
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access public
*/
class nusoap_fault extends nusoap_base {
        /**
         * The fault code (client|server)
         * @var string
         * @access private
         */
        var $faultcode;
        /**
         * The fault actor
         * @var string
         * @access private
```

```
     */
    var $faultactor;
    /**
     * The fault string, a description of the fault
     * @var string
     * @access private
     */
    var $faultstring;
    /**
     * The fault detail, typically a string or array of string
     * @var mixed
     * @access private
     */
    var $faultdetail;

    /**
    * constructor
*
* @param string $faultcode (SOAP-ENV:Client | SOAP-ENV:Server)
* @param string $faultactor only used when msg routed between multiple actors
```

```
* @param string $faultstring human readable error message
* @param mixed $faultdetail detail, typically a string or array of string
        */
        function nusoap_fault($faultcode,$faultactor='',$faultstring='',$faultdetail=''){
                parent::nusoap_base();
                $this->faultcode = $faultcode;
                $this->faultactor = $faultactor;
                $this->faultstring = $faultstring;
                $this->faultdetail = $faultdetail;
        }

        /**
        * serialize a fault
        *
        * @return       string   The serialization of the fault instance.
        * @access   public
        */
        function serialize(){
                $ns_string = '';
                foreach($this->namespaces as $k => $v){
```

```
                $ns_string .= "\n  xmlns:$k=\"$v\"";
            }
            $return_msg =
                '<?xml version="1.0" encoding="'.$this->soap_defencoding.'"?>'.
                '<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/".$ns_string.">\n".
                    '<SOAP-ENV:Body>'.
                    '<SOAP-ENV:Fault>'.
                        $this->serialize_val($this->faultcode, 'faultcode').
                        $this->serialize_val($this->faultactor, 'faultactor').
                        $this->serialize_val($this->faultstring, 'faultstring').
                        $this->serialize_val($this->faultdetail, 'detail').
                    '</SOAP-ENV:Fault>'.
                    '</SOAP-ENV:Body>'.
                '</SOAP-ENV:Envelope>';
            return $return_msg;
        }
}
```

```
/**
 * Backward compatibility
 */
class soap_fault extends nusoap_fault {
}

?><?php

/**
* parses an XML Schema, allows access to it's data, other utility methods.
* imperfect, no validation... yet, but quite functional.
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
```

```
class nusoap_xmlschema extends nusoap_base  {

    // files
    var $schema = '';
    var $xml = '';
    // namespaces
    var $enclosingNamespaces;
    // schema info
    var $schemaInfo = array();
    var $schemaTargetNamespace = '';
    // types, elements, attributes defined by the schema
    var $attributes = array();
    var $complexTypes = array();
    var $complexTypeStack = array();
    var $currentComplexType = null;
    var $elements = array();
    var $elementStack = array();
    var $currentElement = null;
    var $simpleTypes = array();
    var $simpleTypeStack = array();
```

```
var $currentSimpleType = null;
// imports
var $imports = array();
// parser vars
var $parser;
var $position = 0;
var $depth = 0;
var $depth_array = array();
var $message = array();
var $defaultNamespace = array();

/**
* constructor
*
* @param    string $schema schema document URI
* @param    string $xml xml document URI
* @param       string $namespaces namespaces defined in enclosing XML
* @access   public
*/
function nusoap_xmlschema($schema='',$xml='',$namespaces=array()){
```

```
        parent::nusoap_base();
        $this->debug('nusoap_xmlschema class instantiated, inside
constructor');
        // files
        $this->schema = $schema;
        $this->xml = $xml;

        // namespaces
        $this->enclosingNamespaces = $namespaces;
        $this->namespaces = array_merge($this->namespaces, $namespaces);

        // parse schema file
        if($schema != ''){
                $this->debug('initial schema file: '.$schema);
                $this->parseFile($schema, 'schema');
        }

        // parse xml file
        if($xml != ''){
                $this->debug('initial xml file: '.$xml);
```

```
                $this->parseFile($xml, 'xml');
            }

        }

/**
* parse an XML file
*
* @param string $xml path/URL to XML file
* @param string $type (schema | xml)
    * @return boolean
* @access public
*/
    function parseFile($xml,$type){
        // parse xml file
        if($xml != ""){
            $xmlStr = @join("",@file($xml));
            if($xmlStr == ""){
                $msg = 'Error reading XML from '.$xml;
                $this->setError($msg);
```

```
                $this->debug($msg);
            return false;
            } else {
                $this->debug("parsing $xml");
                $this->parseString($xmlStr,$type);
                $this->debug("done parsing $xml");
            return true;
            }
        }
        return false;
    }

    /**
    * parse an XML string
    *
    * @param    string $xml path or URL
* @param   string $type (schema|xml)
    * @access   private
    */
    function parseString($xml,$type){
```

```
// parse xml string
if($xml != ""){

// Create an XML parser.
$this->parser = xml_parser_create();
// Set the options for parsing the XML data.
xml_parser_set_option($this->parser,
XML_OPTION_CASE_FOLDING, 0);

// Set the object for the parser.
xml_set_object($this->parser, $this);

// Set the element handlers for the parser.
if($type == "schema"){
xml_set_element_handler($this->parser,
'schemaStartElement','schemaEndElement');
xml_set_character_data_handler($this-
>parser,'schemaCharacterData');
} elseif($type == "xml"){
xml_set_element_handler($this->parser,
'xmlStartElement','xmlEndElement');
```

```
                xml_set_character_data_handler($this-
>parser,'xmlCharacterData');
                }

            // Parse the XML file.
            if(!xml_parse($this->parser,$xml,true)){
                // Display an error message.
                    $errstr = sprintf('XML error parsing XML schema on
line %d: %s',
                    xml_get_current_line_number($this->parser),
                    xml_error_string(xml_get_error_code($this->parser))
                    );
                    $this->debug($errstr);
                    $this->debug("XML payload:\n" . $xml);
                    $this->setError($errstr);
            }

                xml_parser_free($this->parser);
            } else{
                $this->debug('no xml passed to parseString()!!');
                $this->setError('no xml passed to parseString()!!');
```

```
        }
}

/**
 * gets a type name for an unnamed type
 *
 * @param      string   Element name
 * @return     string   A type name for an unnamed type
 * @access     private
 */
function CreateTypeName($ename) {
        $scope = '';
        for ($i = 0; $i < count($this->complexTypeStack); $i++) {
                $scope .= $this->complexTypeStack[$i] . '_';
        }
        return $scope . $ename . '_ContainedType';
}

/**
* start-element handler
```

```
        *
        * @param    string $parser XML parser object
        * @param    string $name element name
        * @param    string $attrs associative array of attributes
        * @access   private
        */
        function schemaStartElement($parser, $name, $attrs) {

                // position in the total number of elements, starting from 0
                $pos = $this->position++;
                $depth = $this->depth++;
                // set self as current value for this depth
                $this->depth_array[$depth] = $pos;
                $this->message[$pos] = array('cdata' => '');
                if ($depth > 0) {
                        $this->defaultNamespace[$pos] = $this-
>defaultNamespace[$this->depth_array[$depth - 1]];
                } else {
                        $this->defaultNamespace[$pos] = false;
                }
```

```
            // get element prefix
            if($prefix = $this->getPrefix($name)){
                    // get unqualified name
                    $name = $this->getLocalPart($name);
            } else {
   $prefix = '';
}

// loop thru attributes, expanding, and registering namespace declarations
if(count($attrs) > 0){
   foreach($attrs as $k => $v){
      // if ns declarations, add to class level array of valid namespaces
                              if(preg_match('/^xmlns/',$k)){
            //$this->xdebug("$k: $v");
            //$this->xdebug('ns_prefix: '.$this->getPrefix($k));
            if($ns_prefix = substr(strrchr($k,':'),1)){
                    //$this->xdebug("Add namespace[$ns_prefix] = $v");
                                            $this->namespaces[$ns_prefix] = $v;
                                    } else {
```

```
                                        $this->defaultNamespace[$pos] = $v;
                                        if (! $this->getPrefixFromNamespace($v)) {
                                            $this->namespaces['ns'.(count($this->namespaces)+1)] = $v;
                                        }
                                    }
                                    if($v == 'http://www.w3.org/2001/XMLSchema' || $v == 'http://www.w3.org/1999/XMLSchema' || $v == 'http://www.w3.org/2000/10/XMLSchema'){
                                        $this->XMLSchemaVersion = $v;
                                        $this->namespaces['xsi'] = $v.'-instance';
                                    }
                                }
        }
        foreach($attrs as $k => $v){
            // expand each attribute
            $k = strpos($k,':') ? $this->expandQname($k) : $k;
            $v = strpos($v,':') ? $this->expandQname($v) : $v;
                $eAttrs[$k] = $v;
        }
```

```
            $attrs = $eAttrs;
        } else {
            $attrs = array();
        }
                // find status, register data
                switch($name){
                    case 'all':                     // (optional) compositor content for a complexType
                    case 'choice':
                    case 'group':
                    case 'sequence':
                        //$this->xdebug("compositor $name for currentComplexType: $this->currentComplexType and currentElement: $this->currentElement");
                        $this->complexTypes[$this->currentComplexType]['compositor'] = $name;
                        //if($name == 'all' || $name == 'sequence'){
                        //    $this->complexTypes[$this->currentComplexType]['phpType'] = 'struct';
                        //}
                    break;
                    case 'attribute':           // complexType attribute
```

```
                //$this->xdebug("parsing attribute $attrs[name] $attrs[ref] of value:
".$attrs['http://schemas.xmlsoap.org/wsdl/:arrayType']);

                $this->xdebug("parsing attribute:");

                $this->appendDebug($this->varDump($attrs));

                        if (!isset($attrs['form'])) {

                                // TODO: handle globals

                                $attrs['form'] = $this-
>schemaInfo['attributeFormDefault'];

                        }

                if (isset($attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'])) {

                                $v =
$attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'];

                                if (!strpos($v, ':')) {

                                        // no namespace in arrayType attribute
value...

                                        if ($this->defaultNamespace[$pos]) {

                                                // ...so use the default


        $attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'] = $this-
>defaultNamespace[$pos] . ':' . $attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'];

                                        }

                                }
```

```
                }
            if(isset($attrs['name'])){
                                    $this->attributes[$attrs['name']] = $attrs;
                                    $aname = $attrs['name'];
                                } elseif(isset($attrs['ref']) && $attrs['ref'] ==
'http://schemas.xmlsoap.org/soap/encoding/:arrayType'){
                                    if
(isset($attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'])) {
                        $aname = $attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'];
                    } else {
                        $aname = '';
                    }
                                } elseif(isset($attrs['ref'])){
                                    $aname = $attrs['ref'];
                $this->attributes[$attrs['ref']] = $attrs;
                                }

                                if($this->currentComplexType){      // This should
*always* be
                                    $this->complexTypes[$this-
>currentComplexType]['attrs'][$aname] = $attrs;
```

```
                }

                // arrayType attribute

        if(isset($attrs['http://schemas.xmlsoap.org/wsdl/:arrayType']) || $this->getLocalPart($aname) == 'arrayType'){

                        $this->complexTypes[$this->currentComplexType]['phpType'] = 'array';

                $prefix = $this->getPrefix($aname);


        if(isset($attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'])){

                                $v = $attrs['http://schemas.xmlsoap.org/wsdl/:arrayType'];

                        } else {

                                $v = '';

                        }

                if(strpos($v,'[,]')){

                   $this->complexTypes[$this->currentComplexType]['multidimensional'] = true;

                }

                $v = substr($v,0,strpos($v,'[')); // clip the []

                if(!strpos($v,':') && isset($this->typemap[$this->XMLSchemaVersion][$v])){
```

```
                $v = $this->XMLSchemaVersion.':'.$v;
            }
            $this->complexTypes[$this->currentComplexType]['arrayType'] = $v;
                    }
                break;
                case 'complexContent':          // (optional) content for a
complexType
                    $this->xdebug("do nothing for element $name");
                break;
                case 'complexType':
                    array_push($this->complexTypeStack, $this-
>currentComplexType);
                    if(isset($attrs['name'])){
                        // TODO: what is the scope of named
complexTypes that appear
                        //      nested within other c complexTypes?
                        $this->xdebug('processing named complexType
'.$attrs['name']);
                        //$this->currentElement = false;
                        $this->currentComplexType = $attrs['name'];
                        $this->complexTypes[$this-
>currentComplexType] = $attrs;
```

```
                                        $this->complexTypes[$this-
>currentComplexType]['typeClass'] = 'complexType';
                                        // This is for constructs like
                                        //           <complexType name="ListOfString"
base="soap:Array">
                                        //               <sequence>
                                        //                   <element name="string"
type="xsd:string"
                                        //                       minOccurs="0"
maxOccurs="unbounded" />
                                        //               </sequence>
                                        //           </complexType>
                                        if(isset($attrs['base']) &&
preg_match('/:Array$/',$attrs['base'])){
                                            $this->xdebug('complexType is unusual
array');
                                            $this->complexTypes[$this-
>currentComplexType]['phpType'] = 'array';
                                        } else {
                                            $this->complexTypes[$this-
>currentComplexType]['phpType'] = 'struct';
                                        }
                                    } else {
```

```
                                        $name = $this->CreateTypeName($this-
>currentElement);
                                        $this->xdebug('processing unnamed
complexType for element ' . $this->currentElement . ' named ' . $name);
                                        $this->currentComplexType = $name;
                                        //$this->currentElement = false;
                                        $this->complexTypes[$this-
>currentComplexType] = $attrs;
                                        $this->complexTypes[$this-
>currentComplexType]['typeClass'] = 'complexType';
                                        // This is for constructs like
                                        //           <complexType name="ListOfString"
base="soap:Array">
                                        //               <sequence>
                                        //                   <element name="string"
type="xsd:string"
                                        //                       minOccurs="0"
maxOccurs="unbounded" />
                                        //               </sequence>
                                        //           </complexType>
                                        if(isset($attrs['base']) &&
preg_match('/:Array$/',$attrs['base'])){
```

```
                                        $this->xdebug('complexType is unusual
array');
                                        $this->complexTypes[$this-
>currentComplexType]['phpType'] = 'array';
                                    } else {
                                        $this->complexTypes[$this-
>currentComplexType]['phpType'] = 'struct';
                                    }
                                }
                                $this->complexTypes[$this-
>currentComplexType]['simpleContent'] = 'false';
                            break;
                            case 'element':
                                array_push($this->elementStack, $this-
>currentElement);
                                if (!isset($attrs['form'])) {
                                    if ($this->currentComplexType) {
                                        $attrs['form'] = $this-
>schemaInfo['elementFormDefault'];
                                    } else {
                                        // global
                                        $attrs['form'] = 'qualified';
```

```
                }
            }
            if(isset($attrs['type'])){
                $this->xdebug("processing typed element
".$attrs['name']." of type ".$attrs['type']);
                if (! $this->getPrefix($attrs['type'])) {
                    if ($this->defaultNamespace[$pos]) {
                        $attrs['type'] = $this-
>defaultNamespace[$pos] . ':' . $attrs['type'];
                        $this->xdebug('used default
namespace to make type ' . $attrs['type']);
                    }
                }
                // This is for constructs like
                //           <complexType name="ListOfString"
base="soap:Array">
                //                <sequence>
                //                   <element name="string"
type="xsd:string"
                //                      minOccurs="0"
maxOccurs="unbounded" />
                //                </sequence>
```

```
                                        //          </complexType>
                                        if ($this->currentComplexType && $this->complexTypes[$this->currentComplexType]['phpType'] == 'array') {
                                            $this->xdebug('arrayType for unusual array is ' . $attrs['type']);
                                            $this->complexTypes[$this->currentComplexType]['arrayType'] = $attrs['type'];
                                        }
                                        $this->currentElement = $attrs['name'];
                                        $ename = $attrs['name'];
                                    } elseif(isset($attrs['ref'])){
                                        $this->xdebug("processing element as ref to ".$attrs['ref']);
                                        $this->currentElement = "ref to ".$attrs['ref'];
                                        $ename = $this->getLocalPart($attrs['ref']);
                                    } else {
                                        $type = $this->CreateTypeName($this->currentComplexType . '_' . $attrs['name']);
                                        $this->xdebug("processing untyped element " . $attrs['name'] . ' type ' . $type);
                                        $this->currentElement = $attrs['name'];
```

```
                $attrs['type'] = $this->schemaTargetNamespace .
':' . $type;
                $ename = $attrs['name'];
            }
            if (isset($ename) && $this->currentComplexType) {
                $this->xdebug("add element $ename to
complexType $this->currentComplexType");
                $this->complexTypes[$this-
>currentComplexType]['elements'][$ename] = $attrs;
            } elseif (!isset($attrs['ref'])) {
                $this->xdebug("add element $ename to elements
array");
                $this->elements[ $attrs['name'] ] = $attrs;
                $this->elements[ $attrs['name'] ]['typeClass'] =
'element';
            }
        break;
        case 'enumeration':     //          restriction value list member
            $this->xdebug('enumeration ' . $attrs['value']);
            if ($this->currentSimpleType) {
                $this->simpleTypes[$this-
>currentSimpleType]['enumeration'][] = $attrs['value'];
```

```
                } elseif ($this->currentComplexType) {
                    $this->complexTypes[$this->currentComplexType]['enumeration'][] = $attrs['value'];
                }
            break;
            case 'extension':        // simpleContent or complexContent type extension
                $this->xdebug('extension ' . $attrs['base']);
                if ($this->currentComplexType) {
                    $ns = $this->getPrefix($attrs['base']);
                    if ($ns == '') {
                        $this->complexTypes[$this->currentComplexType]['extensionBase'] = $this->schemaTargetNamespace . ':' . $attrs['base'];
                    } else {
                        $this->complexTypes[$this->currentComplexType]['extensionBase'] = $attrs['base'];
                    }
                } else {
                    $this->xdebug('no current complexType to set extensionBase');
                }
```

```
                    break;
                    case 'import':
                        if (isset($attrs['schemaLocation'])) {
                                $this->xdebug('import namespace ' .
$attrs['namespace'] . ' from ' . $attrs['schemaLocation']);
                $this->imports[$attrs['namespace']][] = array('location' =>
$attrs['schemaLocation'], 'loaded' => false);
                            } else {
                                $this->xdebug('import namespace ' .
$attrs['namespace']);
                $this->imports[$attrs['namespace']][] = array('location' => '', 'loaded' =>
true);
                                if (! $this-
>getPrefixFromNamespace($attrs['namespace'])) {
                                    $this->namespaces['ns'.(count($this-
>namespaces)+1)] = $attrs['namespace'];
                                }
                            }
                    break;
                    case 'include':
                        if (isset($attrs['schemaLocation'])) {
```

```
                                $this->xdebug('include into namespace ' . $this->schemaTargetNamespace . ' from ' . $attrs['schemaLocation']);
                $this->imports[$this->schemaTargetNamespace][] = array('location' => $attrs['schemaLocation'], 'loaded' => false);
                            } else {
                                $this->xdebug('ignoring invalid XML Schema construct: include without schemaLocation attribute');
                            }
                        break;
                        case 'list':        // simpleType value list
                            $this->xdebug("do nothing for element $name");
                        break;
                        case 'restriction':         // simpleType, simpleContent or complexContent value restriction
                            $this->xdebug('restriction ' . $attrs['base']);
                            if($this->currentSimpleType){
                                $this->simpleTypes[$this->currentSimpleType]['type'] = $attrs['base'];
                            } elseif($this->currentComplexType){
                                $this->complexTypes[$this->currentComplexType]['restrictionBase'] = $attrs['base'];
                                if(strstr($attrs['base'],':') == ':Array'){
```

```
                                        $this->complexTypes[$this->currentComplexType]['phpType'] = 'array';
                                    }
                                }
                            break;
                            case 'schema':
                                $this->schemaInfo = $attrs;
                                $this->schemaInfo['schemaVersion'] = $this->getNamespaceFromPrefix($prefix);
                                if (isset($attrs['targetNamespace'])) {
                                    $this->schemaTargetNamespace = $attrs['targetNamespace'];
                                }
                                if (!isset($attrs['elementFormDefault'])) {
                                    $this->schemaInfo['elementFormDefault'] = 'unqualified';
                                }
                                if (!isset($attrs['attributeFormDefault'])) {
                                    $this->schemaInfo['attributeFormDefault'] = 'unqualified';
                                }
                            break;
```

```
                case 'simpleContent':  // (optional) content for a complexType
                        if ($this->currentComplexType) {     // This should
*always* be
                                $this->complexTypes[$this-
>currentComplexType]['simpleContent'] = 'true';
                        } else {
                                $this->xdebug("do nothing for element $name
because there is no current complexType");
                        }
                break;
                case 'simpleType':
                        array_push($this->simpleTypeStack, $this-
>currentSimpleType);
                        if(isset($attrs['name'])){
                                $this->xdebug("processing simpleType for name
" . $attrs['name']);
                                $this->currentSimpleType = $attrs['name'];
                                $this->simpleTypes[ $attrs['name'] ] = $attrs;
                                $this->simpleTypes[ $attrs['name'] ]['typeClass']
= 'simpleType';
                                $this->simpleTypes[ $attrs['name'] ]['phpType']
= 'scalar';
```

```
                } else {
                    $name = $this->CreateTypeName($this->currentComplexType . '_' . $this->currentElement);
                    $this->xdebug('processing unnamed simpleType for element ' . $this->currentElement . ' named ' . $name);
                    $this->currentSimpleType = $name;
                    //$this->currentElement = false;
                    $this->simpleTypes[$this->currentSimpleType] = $attrs;
                    $this->simpleTypes[$this->currentSimpleType]['phpType'] = 'scalar';
                }
            break;
            case 'union':    // simpleType type list
                $this->xdebug("do nothing for element $name");
            break;
            default:
                $this->xdebug("do not have any logic to process element $name");
        }
    }
```

```
        /**
        * end-element handler
        *
        * @param    string $parser XML parser object
        * @param    string $name element name
        * @access   private
        */
        function schemaEndElement($parser, $name) {
                // bring depth down a notch
                $this->depth--;
                // position of current element is equal to the last value left in
depth_array for my depth
                if(isset($this->depth_array[$this->depth])){
          $pos = $this->depth_array[$this->depth];
       }
                // get element prefix
                if ($prefix = $this->getPrefix($name)){
                        // get unqualified name
                        $name = $this->getLocalPart($name);
                } else {
```

```
        $prefix = '';
    }
            // move on...
            if($name == 'complexType'){
                $this->xdebug('done processing complexType ' . ($this->currentComplexType ? $this->currentComplexType : '(unknown)'));
                $this->xdebug($this->varDump($this->complexTypes[$this->currentComplexType]));
                $this->currentComplexType = array_pop($this->complexTypeStack);
                //$this->currentElement = false;
            }
            if($name == 'element'){
                $this->xdebug('done processing element ' . ($this->currentElement ? $this->currentElement : '(unknown)'));
                $this->currentElement = array_pop($this->elementStack);
            }
            if($name == 'simpleType'){
                $this->xdebug('done processing simpleType ' . ($this->currentSimpleType ? $this->currentSimpleType : '(unknown)'));
                $this->xdebug($this->varDump($this->simpleTypes[$this->currentSimpleType]));
```

```
                        $this->currentSimpleType = array_pop($this->simpleTypeStack);
                }
        }

        /**
        * element content handler
        *
        * @param    string $parser XML parser object
        * @param    string $data element content
        * @access   private
        */
        function schemaCharacterData($parser, $data){
                $pos = $this->depth_array[$this->depth - 1];
                $this->message[$pos]['cdata'] .= $data;
        }

        /**
        * serialize the schema
        *
```

```
    * @access   public
    */
    function serializeSchema(){

        $schemaPrefix = $this->getPrefixFromNamespace($this->XMLSchemaVersion);
        $xml = '';
        // imports
        if (sizeof($this->imports) > 0) {
            foreach($this->imports as $ns => $list) {
                foreach ($list as $ii) {
                    if ($ii['location'] != '') {
                        $xml .= ' <$schemaPrefix:import location="' . $ii['location'] . '" namespace="' . $ns . '" />\n';
                    } else {
                        $xml .= ' <$schemaPrefix:import namespace="' . $ns . '" />\n';
                    }
                }
            }
        }
```

```
                // complex types
                foreach($this->complexTypes as $typeName => $attrs){
                        $contentStr = '';
                        // serialize child elements
                        if(isset($attrs['elements']) && (count($attrs['elements']) > 0)){
                                foreach($attrs['elements'] as $element => $eParts){
                                        if(isset($eParts['ref'])){
                                                $contentStr .= "
<$schemaPrefix:element ref=\"$element\"/>\n";
                                        } else {
                                                $contentStr .= "
<$schemaPrefix:element name=\"$element\" type=\"" . $this-
>contractQName($eParts['type']) . "\"";
                                                foreach ($eParts as $aName => $aValue)
{
                                                        // handle, e.g., abstract, default,
form, minOccurs, maxOccurs, nillable
                                                        if ($aName != 'name' && $aName
!= 'type') {
                                                                $contentStr .= "
$aName=\"$aValue\"";
                                                        }
```

```
                                        }
                                        $contentStr .= "/>\n";
                                    }
                                }
                                // compositor wraps elements
                                if (isset($attrs['compositor']) && ($attrs['compositor'] !=
")) {
                                    $contentStr = "
<$schemaPrefix:$attrs[compositor]>\n".$contentStr."
</$schemaPrefix:$attrs[compositor]>\n";
                                }
                            }
                            // attributes
                            if(isset($attrs['attrs']) && (count($attrs['attrs']) >= 1)){
                                foreach($attrs['attrs'] as $attr => $aParts){
                                    $contentStr .= "    <$schemaPrefix:attribute";
                                    foreach ($aParts as $a => $v) {
                                        if ($a == 'ref' || $a == 'type') {
                                            $contentStr .= " $a=\"".$this-
>contractQName($v)."";
                                        } elseif ($a ==
'http://schemas.xmlsoap.org/wsdl/:arrayType') {
```

```
                                        $this->usedNamespaces['wsdl'] =
$this->namespaces['wsdl'];
                                        $contentStr .= '
wsdl:arrayType="'.$this->contractQName($v).""';
                                    } else {
                                        $contentStr .= " $a=\"$v\"";
                                    }
                                }
                                $contentStr .= "/>\n";
                            }
                        }
                        // if restriction
                        if (isset($attrs['restrictionBase']) && $attrs['restrictionBase'] !=
''){
                            $contentStr = "   <$schemaPrefix:restriction
base=\"".$this->contractQName($attrs['restrictionBase'])."\">\n".$contentStr."
</$schemaPrefix:restriction>\n";
                            // complex or simple content
                            if ((isset($attrs['elements']) && count($attrs['elements'])
> 0) || (isset($attrs['attrs']) && count($attrs['attrs']) > 0)){
                                $contentStr = "
<$schemaPrefix:complexContent>\n".$contentStr."
</$schemaPrefix:complexContent>\n";
```

```
                    }
                }
                // finalize complex type
                if($contentStr != ''){
                    $contentStr = " <$schemaPrefix:complexType
name=\"$typeName\">\n".$contentStr." </$schemaPrefix:complexType>\n";
                } else {
                    $contentStr = " <$schemaPrefix:complexType
name=\"$typeName\"/>\n";
                }
                $xml .= $contentStr;
            }
            // simple types
            if(isset($this->simpleTypes) && count($this->simpleTypes) > 0){
                foreach($this->simpleTypes as $typeName => $eParts){
                    $xml .= " <$schemaPrefix:simpleType
name=\"$typeName\">\n  <$schemaPrefix:restriction base=\"".$this-
>contractQName($eParts['type'])."\">\n";
                    if (isset($eParts['enumeration'])) {
                        foreach ($eParts['enumeration'] as $e) {
                            $xml .= "  <$schemaPrefix:enumeration
value=\"$e\"/>\n";
```

```
                                        }

                                    }

                                    $xml .= "  </$schemaPrefix:restriction>\n
</$schemaPrefix:simpleType>";

                                }

                            }

                            // elements

                            if(isset($this->elements) && count($this->elements) > 0){

                                foreach($this->elements as $element => $eParts){

                                    $xml .= " <$schemaPrefix:element name=\"$element\"
type=\"".$this->contractQName($eParts['type'])."\"/>\n";

                                }

                            }

                            // attributes

                            if(isset($this->attributes) && count($this->attributes) > 0){

                                foreach($this->attributes as $attr => $aParts){

                                    $xml .= " <$schemaPrefix:attribute name=\"$attr\"
type=\"".$this->contractQName($aParts['type'])."\"\n/>";

                                }

                            }

                            // finish 'er up
```

```
        $attr = '';
        foreach ($this->schemaInfo as $k => $v) {
            if ($k == 'elementFormDefault' || $k == 'attributeFormDefault')
{
                $attr .= " $k=\"$v\"";
            }
        }
        $el = "<$schemaPrefix:schema$attr targetNamespace=\"$this-
>schemaTargetNamespace\"\n";
        foreach (array_diff($this->usedNamespaces, $this-
>enclosingNamespaces) as $nsp => $ns) {
            $el .= " xmlns:$nsp=\"$ns\"";
        }
        $xml = $el . ">\n".$xml."</$schemaPrefix:schema>\n";
        return $xml;
    }

    /**
    * adds debug data to the clas level debug string
    *
    * @param    string $string debug data
```

```
        * @access   private
        */
        function xdebug($string){
                $this->debug('<' . $this->schemaTargetNamespace . '> '.$string);
        }

/**
* get the PHP type of a user defined type in the schema
* PHP type is kind of a misnomer since it actually returns 'struct' for assoc. arrays
* returns false if no type exists, or not w/ the given namespace
* else returns a string that is either a native php type, or 'struct'
*
* @param string $type name of defined type
* @param string $ns namespace of type
* @return mixed
* @access public
* @deprecated
*/
        function getPHPType($type,$ns){
                if(isset($this->typemap[$ns][$type])){
```

```
                    //print "found type '$type' and ns $ns in typemap<br>";
                    return $this->typemap[$ns][$type];
                } elseif(isset($this->complexTypes[$type])){
                    //print "getting type '$type' and ns $ns from complexTypes array<br>";
                    return $this->complexTypes[$type]['phpType'];
                }
                return false;
        }

        /**
    * returns an associative array of information about a given type
    * returns false if no type exists by the given name
    *
        *       For a complexType typeDef = array(
        *       'restrictionBase' => '',
        *       'phpType' => '',
        *       'compositor' => '(sequence|all)',
        *       'elements' => array(), // refs to elements array
        *       'attrs' => array() // refs to attributes array
```

```
        *       ... and so on (see addComplexType)
        *       )
        *
        *   For simpleType or element, the array has different keys.
*
* @param string $type
* @return mixed
* @access public
* @see addComplexType
* @see addSimpleType
* @see addElement
*/
        function getTypeDef($type){
                //$this->debug("in getTypeDef for type $type");
                if (substr($type, -1) == '^') {
                        $is_element = 1;
                        $type = substr($type, 0, -1);
                } else {
                        $is_element = 0;
                }
```

```
            if((! $is_element) && isset($this->complexTypes[$type])){
                $this->xdebug("in getTypeDef, found complexType $type");
                return $this->complexTypes[$type];
            } elseif((! $is_element) && isset($this->simpleTypes[$type])){
                $this->xdebug("in getTypeDef, found simpleType $type");
                if (!isset($this->simpleTypes[$type]['phpType'])) {
                    // get info for type to tack onto the simple type
                    // TODO: can this ever really apply (i.e. what is a
simpleType really?)
                    $uqType = substr($this->simpleTypes[$type]['type'],
strrpos($this->simpleTypes[$type]['type'], ':') + 1);
                    $ns = substr($this->simpleTypes[$type]['type'], 0,
strrpos($this->simpleTypes[$type]['type'], ':'));
                    $etype = $this->getTypeDef($uqType);
                    if ($etype) {
                        $this->xdebug("in getTypeDef, found type for
simpleType $type:");
                        $this->xdebug($this->varDump($etype));
                        if (isset($etype['phpType'])) {
                            $this->simpleTypes[$type]['phpType'] =
$etype['phpType'];
```

```
                                        }
                                        if (isset($etype['elements'])) {
                                                $this->simpleTypes[$type]['elements'] =
$etype['elements'];
                                        }
                                }
                        }
                        return $this->simpleTypes[$type];
                } elseif(isset($this->elements[$type])){
                        $this->xdebug("in getTypeDef, found element $type");
                        if (!isset($this->elements[$type]['phpType'])) {
                                // get info for type to tack onto the element
                                $uqType = substr($this->elements[$type]['type'],
strrpos($this->elements[$type]['type'], ':') + 1);
                                $ns = substr($this->elements[$type]['type'], 0,
strrpos($this->elements[$type]['type'], ':'));
                                $etype = $this->getTypeDef($uqType);
                                if ($etype) {
                                        $this->xdebug("in getTypeDef, found type for
element $type:");
                                        $this->xdebug($this->varDump($etype));
```

```
                                        if (isset($etype['phpType'])) {
                                                $this->elements[$type]['phpType'] =
$etype['phpType'];
                                        }
                                        if (isset($etype['elements'])) {
                                                $this->elements[$type]['elements'] =
$etype['elements'];
                                        }
                                        if (isset($etype['extensionBase'])) {
                                                $this->elements[$type]['extensionBase']
= $etype['extensionBase'];
                                        }
                                } elseif ($ns ==
'http://www.w3.org/2001/XMLSchema') {
                                        $this->xdebug("in getTypeDef, element $type is
an XSD type");
                                        $this->elements[$type]['phpType'] = 'scalar';
                                }
                        }
                        return $this->elements[$type];
                } elseif(isset($this->attributes[$type])){
                        $this->xdebug("in getTypeDef, found attribute $type");
```

```
                return $this->attributes[$type];
            } elseif (preg_match('/_ContainedType$/', $type)) {
                $this->xdebug("in getTypeDef, have an untyped element $type");
                $typeDef['typeClass'] = 'simpleType';
                $typeDef['phpType'] = 'scalar';
                $typeDef['type'] = 'http://www.w3.org/2001/XMLSchema:string';
                return $typeDef;
            }
            $this->xdebug("in getTypeDef, did not find $type");
            return false;
        }

        /**
     * returns a sample serialization of a given type, or false if no type by the given name
     *
     * @param string $type name of type
     * @return mixed
     * @access public
```

```
* @deprecated
*/
function serializeTypeDef($type){
    //print "in sTD() for type $type<br>";
    if($typeDef = $this->getTypeDef($type)){
        $str .= '<'.$type;
      if(is_array($typeDef['attrs'])){
        foreach($typeDef['attrs'] as $attName => $data){
          $str .= " $attName=\"{type = ".$data['type']."}\"";
        }
      }
      $str .= " xmlns=\"".$this->schema['targetNamespace']."\"";
      if(count($typeDef['elements']) > 0){
        $str .= ">";
        foreach($typeDef['elements'] as $element => $eData){
          $str .= $this->serializeTypeDef($element);
        }
        $str .= "</$type>";
      } elseif($typeDef['typeClass'] == 'element') {
        $str .= "></$type>";
```

```
            } else {
                $str .= "/>";
            }
                    return $str;
        }
        return false;
}

/**
* returns HTML form elements that allow a user
* to enter values for creating an instance of the given type.
*
* @param string $name name for type instance
* @param string $type name of type
* @return string
* @access public
* @deprecated
        */
        function typeToForm($name,$type){
                // get typedef
```

```
                if($typeDef = $this->getTypeDef($type)){
                        // if struct
                        if($typeDef['phpType'] == 'struct'){
                                $buffer .= '<table>';
                                foreach($typeDef['elements'] as $child => $childDef){
                                        $buffer .= "
                                        <tr><td align='right'>$childDef[name] (type:
".$this->getLocalPart($childDef['type'])."):</td>
                                        <td><input type='text'
name='parameters[".$name."][$childDef[name]]'></td></tr>";
                                }
                                $buffer .= '</table>';
                        // if array
                        } elseif($typeDef['phpType'] == 'array'){
                                $buffer .= '<table>';
                                for($i=0;$i < 3; $i++){
                                        $buffer .= "
                                        <tr><td align='right'>array item (type:
$typeDef[arrayType]):</td>
                                        <td><input type='text'
name='parameters[".$name."][]'></td></tr>";
```

```
                    }
                    $buffer .= '</table>';
                // if scalar
                } else {
                    $buffer .= "<input type='text'
name='parameters[$name]'>";
                }
            } else {
                $buffer .= "<input type='text' name='parameters[$name]'>";
            }
            return $buffer;
        }

        /**
        * adds a complex type to the schema
        *
        * example: array
        *
        * addType(
        *      'ArrayOfstring',
```

```
*          'complexType',
*          'array',
*          '',
*          'SOAP-ENC:Array',
*          array('ref'=>'SOAP-ENC:arrayType','wsdl:arrayType'=>'string[]'),
*          'xsd:string'
* );
*
* example: PHP associative array ( SOAP Struct )
*
* addType(
*          'SOAPStruct',
*          'complexType',
*          'struct',
*          'all',
*          array('myVar'=> array('name'=>'myVar','type'=>'string')
* );
*
* @param name
* @param typeClass (complexType|simpleType|attribute)
```

```
        * @param phpType: currently supported are array and struct (php assoc array)
        * @param compositor (all|sequence|choice)
        * @param restrictionBase namespace:name
(http://schemas.xmlsoap.org/soap/encoding/:Array)
        * @param elements = array ( name = array(name=>'',type=>'') )
        * @param attrs = array(
        *       array(
        *               'ref' =>
"http://schemas.xmlsoap.org/soap/encoding/:arrayType",
        *               "http://schemas.xmlsoap.org/wsdl/:arrayType" => "string[]"
        *       )
        * )
        * @param arrayType: namespace:name
(http://www.w3.org/2001/XMLSchema:string)
        * @access public
        * @see getTypeDef
        */
        function
addComplexType($name,$typeClass='complexType',$phpType='array',$compositor=
'',$restrictionBase='',$elements=array(),$attrs=array(),$arrayType=''){
                $this->complexTypes[$name] = array(
            'name'              => $name,
```

```
            'typeClass'   => $typeClass,
            'phpType'    => $phpType,
                  'compositor'=> $compositor,
            'restrictionBase' => $restrictionBase,
                  'elements'       => $elements,
            'attrs'                  => $attrs,
            'arrayType'  => $arrayType
                  );

                  $this->xdebug("addComplexType $name:");
                  $this->appendDebug($this->varDump($this->complexTypes[$name]));
            }

            /**
            * adds a simple type to the schema
            *
            * @param string $name
            * @param string $restrictionBase namespace:name (http://schemas.xmlsoap.org/soap/encoding/:Array)
            * @param string $typeClass (should always be simpleType)
```

```
        * @param string $phpType (should always be scalar)
        * @param array $enumeration array of values
        * @access public
        * @see nusoap_xmlschema
        * @see getTypeDef
        */
        function addSimpleType($name, $restrictionBase='',
$typeClass='simpleType', $phpType='scalar', $enumeration=array()) {
                $this->simpleTypes[$name] = array(
           'name'                   => $name,
           'typeClass'          => $typeClass,
           'phpType'            => $phpType,
           'type'                   => $restrictionBase,
           'enumeration'        => $enumeration
                );

                $this->xdebug("addSimpleType $name:");
                $this->appendDebug($this->varDump($this->simpleTypes[$name]));
        }
```

```
        /**
        * adds an element to the schema
        *
        * @param array $attrs attributes that must include name and type
        * @see nusoap_xmlschema
        * @access public
        */
        function addElement($attrs) {
                if (! $this->getPrefix($attrs['type'])) {
                        $attrs['type'] = $this->schemaTargetNamespace . ':' .
$attrs['type'];
                }
                $this->elements[ $attrs['name'] ] = $attrs;
                $this->elements[ $attrs['name'] ]['typeClass'] = 'element';

                $this->xdebug("addElement " . $attrs['name']);
                $this->appendDebug($this->varDump($this->elements[ $attrs['name']
]));
        }
}
```

```
/**
 * Backward compatibility
 */
class XMLSchema extends nusoap_xmlschema {
}

?><?php

/**
* For creating serializable abstractions of native PHP types.  This class
* allows element name/namespace, XSD type, and XML attributes to be
* associated with a value.  This is extremely useful when WSDL is not
* used, but is also useful when WSDL is used with polymorphic types, including
* xsd:anyType and user-defined types.
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class soapval extends nusoap_base {
```

```
/**
 * The XML element name
 *
 * @var string
 * @access private
 */
var $name;
/**
 * The XML type name (string or false)
 *
 * @var mixed
 * @access private
 */
var $type;
/**
 * The PHP value
 *
 * @var mixed
 * @access private
 */
```

```
var $value;
/**
 * The XML element namespace (string or false)
 *
 * @var mixed
 * @access private
 */
var $element_ns;
/**
 * The XML type namespace (string or false)
 *
 * @var mixed
 * @access private
 */
var $type_ns;
/**
 * The XML element attributes (array or false)
 *
 * @var mixed
 * @access private
```

```
     */
    var $attributes;

    /**
    * constructor
    *
    * @param    string $name optional name
    * @param    mixed $type optional type name
    * @param      mixed $value optional value
    * @param      mixed $element_ns optional namespace of value
    * @param      mixed $type_ns optional namespace of type
    * @param      mixed $attributes associative array of attributes to add to element serialization
    * @access   public
    */
    function soapval($name='soapval',$type=false,$value=-1,$element_ns=false,$type_ns=false,$attributes=false) {
        parent::nusoap_base();
        $this->name = $name;
        $this->type = $type;
        $this->value = $value;
```

```
            $this->element_ns = $element_ns;
            $this->type_ns = $type_ns;
            $this->attributes = $attributes;
    }

        /**
        * return serialized value
        *
        * @param       string $use The WSDL use value (encoded|literal)
        * @return      string XML data
        * @access   public
        */
        function serialize($use='encoded') {
            return $this->serialize_val($this->value, $this->name, $this->type,
$this->element_ns, $this->type_ns, $this->attributes, $use, true);
    }

        /**
        * decodes a soapval object into a PHP native type
        *
```

```
        * @return       mixed
        * @access   public
        */
        function decode(){
                return $this->value;
        }
}




?><?php




/**
* transport class for sending/receiving data via HTTP and HTTPS
* NOTE: PHP must be compiled with the CURL extension for HTTPS support
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
```

```
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access public
*/
class soap_transport_http extends nusoap_base {

        var $url = '';
        var $uri = '';
        var $digest_uri = '';
        var $scheme = '';
        var $host = '';
        var $port = '';
        var $path = '';
        var $request_method = 'POST';
        var $protocol_version = '1.0';
        var $encoding = '';
        var $outgoing_headers = array();
        var $incoming_headers = array();
        var $incoming_cookies = array();
        var $outgoing_payload = '';
        var $incoming_payload = '';
```

```
        var $response_status_line;     // HTTP response status line
        var $useSOAPAction = true;
        var $persistentConnection = false;
        var $ch = false;          // cURL handle
        var $ch_options = array();     // cURL custom options
        var $use_curl = false;            // force cURL use
        var $proxy = null;                        // proxy information (associative array)
        var $username = '';
        var $password = '';
        var $authtype = '';
        var $digestRequest = array();
        var $certRequest = array();    // keys must be cainfofile (optional), sslcertfile,
sslkeyfile, passphrase, certpassword (optional), verifypeer (optional), verifyhost
(optional)
                                                    // cainfofile: certificate
authority file, e.g. '$pathToPemFiles/rootca.pem'
                                                    // sslcertfile: SSL
certificate file, e.g. '$pathToPemFiles/mycert.pem'
                                                    // sslkeyfile: SSL key file,
e.g. '$pathToPemFiles/mykey.pem'
                                                    // passphrase: SSL key
password/passphrase
```

```
                                                    // certpassword: SSL
certificate password
                                                    // verifypeer: default is 1
                                                    // verifyhost: default is 1
        /**
        * constructor
        *
        * @param string $url The URL to which to connect
        * @param array $curl_options User-specified cURL options
        * @param boolean $use_curl Whether to try to force cURL use
        * @access public
        */
        function soap_transport_http($url, $curl_options = NULL, $use_curl = false){
                parent::nusoap_base();
                $this->debug("ctor url=$url use_curl=$use_curl curl_options:");
                $this->appendDebug($this->varDump($curl_options));
                $this->setURL($url);
                if (is_array($curl_options)) {
                        $this->ch_options = $curl_options;
                }
```

```
        $this->use_curl = $use_curl;
        preg_match('/\$Revisio' . 'n: ([^ ]+)/', $this->revision, $rev);
        $this->setHeader('User-Agent', $this->title.'/'.$this->version.'
('.$rev[1].')');
    }

    /**
    * sets a cURL option
    *
    * @param      mixed $option The cURL option (always integer?)
    * @param      mixed $value The cURL option value
    * @access   private
    */
    function setCurlOption($option, $value) {
        $this->debug("setCurlOption option=$option, value=");
        $this->appendDebug($this->varDump($value));
        curl_setopt($this->ch, $option, $value);
    }

    /**
```

```
* sets an HTTP header
*
* @param string $name The name of the header
* @param string $value The value of the header
* @access private
*/
function setHeader($name, $value) {
        $this->outgoing_headers[$name] = $value;
        $this->debug("set header $name: $value");
}

/**
* unsets an HTTP header
*
* @param string $name The name of the header
* @access private
*/
function unsetHeader($name) {
        if (isset($this->outgoing_headers[$name])) {
                $this->debug("unset header $name");
```

```
                unset($this->outgoing_headers[$name]);
        }
}

/**
* sets the URL to which to connect
*
* @param string $url The URL to which to connect
* @access private
*/
function setURL($url) {
        $this->url = $url;

        $u = parse_url($url);
        foreach($u as $k => $v){
                $this->debug("parsed URL $k = $v");
                $this->$k = $v;
        }

        // add any GET params to path
```

```
        if(isset($u['query']) && $u['query'] != ''){
$this->path .= '?' . $u['query'];
        }

        // set default port
        if(!isset($u['port'])){
                if($u['scheme'] == 'https'){
                        $this->port = 443;
                } else {
                        $this->port = 80;
                }
        }

        $this->uri = $this->path;
        $this->digest_uri = $this->uri;

        // build headers
        if (!isset($u['port'])) {
                $this->setHeader('Host', $this->host);
        } else {
```

```
                $this->setHeader('Host', $this->host.':'.$this->port);
        }

        if (isset($u['user']) && $u['user'] != '') {
                $this->setCredentials(urldecode($u['user']), isset($u['pass']) ?
urldecode($u['pass']) : '');
        }
    }

    /**
    * gets the I/O method to use
    *
    * @return       string   I/O method to use (socket|curl|unknown)
    * @access       private
    */
    function io_method() {
        if ($this->use_curl || ($this->scheme == 'https') || ($this->scheme ==
'http' && $this->authtype == 'ntlm') || ($this->scheme == 'http' && is_array($this-
>proxy) && $this->proxy['authtype'] == 'ntlm'))
                return 'curl';
```

```
            if (($this->scheme == 'http' || $this->scheme == 'ssl') && $this-
>authtype != 'ntlm' && (!is_array($this->proxy) || $this->proxy['authtype'] != 'ntlm'))
                return 'socket';
            return 'unknown';
        }

        /**
        * establish an HTTP connection
        *
        * @param    integer $timeout set connection timeout in seconds
        * @param      integer $response_timeout set response timeout in seconds
        * @return      boolean true if connected, false if not
        * @access   private
        */
        function connect($connection_timeout=0,$response_timeout=30){
            // For PHP 4.3 with OpenSSL, change https scheme to ssl, then treat
like
            // "regular" socket.
            // TODO: disabled for now because OpenSSL must be *compiled* in
(not just
            //       loaded), and until PHP5 stream_get_wrappers is not available.
```

```
//                  if ($this->scheme == 'https') {
//                          if (version_compare(phpversion(), '4.3.0') >= 0) {
//                                  if (extension_loaded('openssl')) {
//                                          $this->scheme = 'ssl';
//                                          $this->debug('Using SSL over OpenSSL');
//                                  }
//                          }
//                  }
                    $this->debug("connect connection_timeout $connection_timeout, response_timeout $response_timeout, scheme $this->scheme, host $this->host, port $this->port");
                if ($this->io_method() == 'socket') {
                    if (!is_array($this->proxy)) {
                            $host = $this->host;
                            $port = $this->port;
                    } else {
                            $host = $this->proxy['host'];
                            $port = $this->proxy['port'];
                    }

                    // use persistent connection
```

```
                if($this->persistentConnection && isset($this->fp) &&
is_resource($this->fp)){
                        if (!feof($this->fp)) {
                                $this->debug('Re-use persistent connection');
                                return true;
                        }
                        fclose($this->fp);
                        $this->debug('Closed persistent connection at EOF');
                }

                // munge host if using OpenSSL
                if ($this->scheme == 'ssl') {
                        $host = 'ssl://' . $host;
                }
                $this->debug('calling fsockopen with host ' . $host . '
connection_timeout ' . $connection_timeout);

                // open socket
                if($connection_timeout > 0){
                        $this->fp = @fsockopen( $host, $this->port, $this->errno,
$this->error_str, $connection_timeout);
```

```
                } else {
                        $this->fp = @fsockopen( $host, $this->port, $this->errno,
$this->error_str);
                }

                // test pointer
                if(!$this->fp) {
                        $msg = 'Couldn\'t open socket connection to server ' . $this-
>url;
                        if ($this->errno) {
                                $msg .= ', Error ('.$this->errno.'): '.$this->error_str;
                        } else {
                                $msg .= ' prior to connect().  This is often a problem
looking up the host name.';
                        }
                        $this->debug($msg);
                        $this->setError($msg);
                        return false;
                }

                // set response timeout
```

```
                $this->debug('set response timeout to ' . $response_timeout);
                socket_set_timeout( $this->fp, $response_timeout);


                $this->debug('socket connected');
                return true;
        } else if ($this->io_method() == 'curl') {
                if (!extension_loaded('curl')) {
//                      $this->setError('cURL Extension, or OpenSSL extension w/ PHP version >= 4.3 is required for HTTPS');
                        $this->setError('The PHP cURL Extension is required for HTTPS or NLTM.  You will need to re-build or update your PHP to include cURL or change php.ini to load the PHP cURL extension.');
                        return false;
                }
                // Avoid warnings when PHP does not have these options
                if (defined('CURLOPT_CONNECTIONTIMEOUT'))
                        $CURLOPT_CONNECTIONTIMEOUT = CURLOPT_CONNECTIONTIMEOUT;
                else
                        $CURLOPT_CONNECTIONTIMEOUT = 78;
                if (defined('CURLOPT_HTTPAUTH'))
```

```
        $CURLOPT_HTTPAUTH = CURLOPT_HTTPAUTH;
else
        $CURLOPT_HTTPAUTH = 107;
if (defined('CURLOPT_PROXYAUTH'))
        $CURLOPT_PROXYAUTH = CURLOPT_PROXYAUTH;
else
        $CURLOPT_PROXYAUTH = 111;
if (defined('CURLAUTH_BASIC'))
        $CURLAUTH_BASIC = CURLAUTH_BASIC;
else
        $CURLAUTH_BASIC = 1;
if (defined('CURLAUTH_DIGEST'))
        $CURLAUTH_DIGEST = CURLAUTH_DIGEST;
else
        $CURLAUTH_DIGEST = 2;
if (defined('CURLAUTH_NTLM'))
        $CURLAUTH_NTLM = CURLAUTH_NTLM;
else
        $CURLAUTH_NTLM = 8;
```

```
        $this->debug('connect using cURL');
        // init CURL
        $this->ch = curl_init();
        // set url
        $hostURL = ($this->port != '') ? "$this->scheme://$this->host:$this->port" : "$this->scheme://$this->host";
        // add path
        $hostURL .= $this->path;
        $this->setCurlOption(CURLOPT_URL, $hostURL);
        // follow location headers (re-directs)
        if (ini_get('safe_mode') || ini_get('open_basedir')) {
            $this->debug('safe_mode or open_basedir set, so do not set CURLOPT_FOLLOWLOCATION');
            $this->debug('safe_mode = ');
            $this->appendDebug($this->varDump(ini_get('safe_mode')));
            $this->debug('open_basedir = ');
            $this->appendDebug($this->varDump(ini_get('open_basedir')));
        } else {
            $this->setCurlOption(CURLOPT_FOLLOWLOCATION, 1);
        }
```

```
                // ask for headers in the response output
                $this->setCurlOption(CURLOPT_HEADER, 1);
                // ask for the response output as the return value
                $this->setCurlOption(CURLOPT_RETURNTRANSFER, 1);
                // encode
                // We manage this ourselves through headers and encoding
//              if(function_exists('gzuncompress')){
//                      $this->setCurlOption(CURLOPT_ENCODING, 'deflate');
//              }
                // persistent connection
                if ($this->persistentConnection) {
                        // I believe the following comment is now bogus, having
applied to
                        // the code when it used CURLOPT_CUSTOMREQUEST to
send the request.
                        // The way we send data, we cannot use persistent connections,
since
                        // there will be some "junk" at the end of our request.
                        //$this->setCurlOption(CURL_HTTP_VERSION_1_1, true);
                        $this->persistentConnection = false;
                        $this->setHeader('Connection', 'close');
```

```
                }
                // set timeouts
                if ($connection_timeout != 0) {
                        $this-
>setCurlOption($CURLOPT_CONNECTIONTIMEOUT, $connection_timeout);
                }
                if ($response_timeout != 0) {
                        $this->setCurlOption(CURLOPT_TIMEOUT,
$response_timeout);
                }

                if ($this->scheme == 'https') {
                        $this->debug('set cURL SSL verify options');
                        // recent versions of cURL turn on peer/host checking by
default,
                        // while PHP binaries are not compiled with a default location
for the
                        // CA cert bundle, so disable peer/host checking.
                        //$this->setCurlOption(CURLOPT_CAINFO, 'f:\php-4.3.2-
win32\extensions\curl-ca-bundle.crt');
                        $this->setCurlOption(CURLOPT_SSL_VERIFYPEER, 0);
                        $this->setCurlOption(CURLOPT_SSL_VERIFYHOST, 0);
```

```
                    // support client certificates (thanks Tobias Boes, Doug
Anarino, Eryan Ariobowo)
                    if ($this->authtype == 'certificate') {
                        $this->debug('set cURL certificate options');
                        if (isset($this->certRequest['cainfofile'])) {
                            $this->setCurlOption(CURLOPT_CAINFO,
$this->certRequest['cainfofile']);
                        }
                        if (isset($this->certRequest['verifypeer'])) {
                            $this-
>setCurlOption(CURLOPT_SSL_VERIFYPEER, $this->certRequest['verifypeer']);
                        } else {
                            $this-
>setCurlOption(CURLOPT_SSL_VERIFYPEER, 1);
                        }
                        if (isset($this->certRequest['verifyhost'])) {
                            $this-
>setCurlOption(CURLOPT_SSL_VERIFYHOST, $this->certRequest['verifyhost']);
                        } else {
                            $this-
>setCurlOption(CURLOPT_SSL_VERIFYHOST, 1);
```

```
                }
                if (isset($this->certRequest['sslcertfile'])) {
                    $this->setCurlOption(CURLOPT_SSLCERT,
$this->certRequest['sslcertfile']);
                }
                if (isset($this->certRequest['sslkeyfile'])) {
                    $this->setCurlOption(CURLOPT_SSLKEY,
$this->certRequest['sslkeyfile']);
                }
                if (isset($this->certRequest['passphrase'])) {
                    $this-
>setCurlOption(CURLOPT_SSLKEYPASSWD, $this->certRequest['passphrase']);
                }
                if (isset($this->certRequest['certpassword'])) {
                    $this-
>setCurlOption(CURLOPT_SSLCERTPASSWD, $this-
>certRequest['certpassword']);
                }
            }
        }
        if ($this->authtype && ($this->authtype != 'certificate')) {
            if ($this->username) {
```

```
                $this->debug('set cURL username/password');
                $this->setCurlOption(CURLOPT_USERPWD, "$this->username:$this->password");
            }
            if ($this->authtype == 'basic') {
                $this->debug('set cURL for Basic authentication');
                $this->setCurlOption($CURLOPT_HTTPAUTH, $CURLAUTH_BASIC);
            }
            if ($this->authtype == 'digest') {
                $this->debug('set cURL for digest authentication');
                $this->setCurlOption($CURLOPT_HTTPAUTH, $CURLAUTH_DIGEST);
            }
            if ($this->authtype == 'ntlm') {
                $this->debug('set cURL for NTLM authentication');
                $this->setCurlOption($CURLOPT_HTTPAUTH, $CURLAUTH_NTLM);
            }
        }
        if (is_array($this->proxy)) {
```

```
                $this->debug('set cURL proxy options');
                if ($this->proxy['port'] != '') {
                        $this->setCurlOption(CURLOPT_PROXY, $this-
>proxy['host'].':'.$this->proxy['port']);
                } else {
                        $this->setCurlOption(CURLOPT_PROXY, $this-
>proxy['host']);
                }
                if ($this->proxy['username'] || $this->proxy['password']) {
                        $this->debug('set cURL proxy authentication options');
                        $this->setCurlOption(CURLOPT_PROXYUSERPWD,
$this->proxy['username'].':'.$this->proxy['password']);
                        if ($this->proxy['authtype'] == 'basic') {
                                $this-
>setCurlOption($CURLOPT_PROXYAUTH, $CURLAUTH_BASIC);
                        }
                        if ($this->proxy['authtype'] == 'ntlm') {
                                $this-
>setCurlOption($CURLOPT_PROXYAUTH, $CURLAUTH_NTLM);
                        }
                }
        }
```

```
            $this->debug('cURL connection set up');
            return true;
        } else {
            $this->setError('Unknown scheme ' . $this->scheme);
            $this->debug('Unknown scheme ' . $this->scheme);
            return false;
        }
    }

    /**
    * sends the SOAP request and gets the SOAP response via HTTP[S]
    *
    * @param    string $data message data
    * @param    integer $timeout set connection timeout in seconds
    * @param    integer $response_timeout set response timeout in seconds
    * @param    array $cookies cookies to send
    * @return   string data
    * @access   public
    */
    function send($data, $timeout=0, $response_timeout=30, $cookies=NULL) {
```

```
$this->debug('entered send() with data of length: '.strlen($data));

$this->tryagain = true;
$tries = 0;
while ($this->tryagain) {
        $this->tryagain = false;
        if ($tries++ < 2) {
                // make connnection
                if (!$this->connect($timeout, $response_timeout)){
                        return false;
                }

                // send request
                if (!$this->sendRequest($data, $cookies)){
                        return false;
                }

                // get response
                $respdata = $this->getResponse();
```

```
                } else {
                        $this->setError("Too many tries to get an OK response
($this->response_status_line)");
                }
        }
        $this->debug('end of send()');
        return $respdata;
    }


    /**
    * sends the SOAP request and gets the SOAP response via HTTPS using
CURL
    *
    * @param    string $data message data
    * @param    integer $timeout set connection timeout in seconds
    * @param      integer $response_timeout set response timeout in seconds
    * @param      array $cookies cookies to send
    * @return       string data
    * @access   public
    * @deprecated
```

```
        */
        function sendHTTPS($data, $timeout=0, $response_timeout=30, $cookies) {
                return $this->send($data, $timeout, $response_timeout, $cookies);
        }

        /**
        * if authenticating, set user credentials here
        *
        * @param    string $username
        * @param    string $password
        * @param       string $authtype (basic|digest|certificate|ntlm)
        * @param       array $digestRequest (keys must be nonce, nc, realm, qop)
        * @param       array $certRequest (keys must be cainfofile (optional),
sslcertfile, sslkeyfile, passphrase, certpassword (optional), verifypeer (optional),
verifyhost (optional): see corresponding options in cURL docs)
        * @access   public
        */
        function setCredentials($username, $password, $authtype = 'basic',
$digestRequest = array(), $certRequest = array()) {
                $this->debug("setCredentials username=$username
authtype=$authtype digestRequest=");
```

```
                $this->appendDebug($this->varDump($digestRequest));
                $this->debug("certRequest=");
                $this->appendDebug($this->varDump($certRequest));
                // cf. RFC 2617
                if ($authtype == 'basic') {
                        $this->setHeader('Authorization', 'Basic
'.base64_encode(str_replace(':',",$username).':'.$password));
                } elseif ($authtype == 'digest') {
                        if (isset($digestRequest['nonce'])) {
                                $digestRequest['nc'] = isset($digestRequest['nc']) ?
$digestRequest['nc']++ : 1;


                                // calculate the Digest hashes (calculate code based on
digest implementation found at:
http://www.rassoc.com/gregr/weblog/stories/2002/07/09/webServicesSecurityHttpDi
gestAuthenticationWithoutActiveDirectory.html)


                                // A1 = unq(username-value) ":" unq(realm-value) ":"
passwd
                                $A1 = $username. ':' . (isset($digestRequest['realm']) ?
$digestRequest['realm'] : '') . ':' . $password;
```

```
// H(A1) = MD5(A1)
$HA1 = md5($A1);

// A2 = Method ":" digest-uri-value
$A2 = $this->request_method . ':' . $this->digest_uri;

// H(A2)
$HA2 =  md5($A2);

// KD(secret, data) = H(concat(secret, ":", data))
// if qop == auth:
// request-digest  = <"> < KD ( H(A1),     unq(nonce-value)
//                         ":" nc-value
//                         ":" unq(cnonce-value)
//                         ":" unq(qop-value)
//                         ":" H(A2)
//                        ) <">
// if qop is missing,
// request-digest  = <"> < KD ( H(A1), unq(nonce-value) ":" H(A2) ) > <">
```

```
                $unhashedDigest = '';
                $nonce = isset($digestRequest['nonce']) ?
$digestRequest['nonce'] : '';
                $cnonce = $nonce;
                if ($digestRequest['qop'] != '') {
                        $unhashedDigest = $HA1 . ':' . $nonce . ':' .
sprintf("%08d", $digestRequest['nc']) . ':' . $cnonce . ':' . $digestRequest['qop'] . ':' .
$HA2;
                } else {
                        $unhashedDigest = $HA1 . ':' . $nonce . ':' .
$HA2;
                }

                $hashedDigest = md5($unhashedDigest);

                $opaque = '';
                if (isset($digestRequest['opaque'])) {
                        $opaque = ', opaque="' .
$digestRequest['opaque'] . '"';
                }
```

```
                                $this->setHeader('Authorization', 'Digest username="' .
$username . '", realm="' . $digestRequest['realm'] . '", nonce="' . $nonce . '", uri="' .
$this->digest_uri . $opaque . '", cnonce="' . $cnonce . '", nc=' . sprintf("%08x",
$digestRequest['nc']) . ', qop="' . $digestRequest['qop'] . '", response="' .
$hashedDigest . '"');
                        }
                } elseif ($authtype == 'certificate') {
                        $this->certRequest = $certRequest;
                        $this->debug('Authorization header not set for certificate');
                } elseif ($authtype == 'ntlm') {
                        // do nothing
                        $this->debug('Authorization header not set for ntlm');
                }
                $this->username = $username;
                $this->password = $password;
                $this->authtype = $authtype;
                $this->digestRequest = $digestRequest;
        }

        /**
        * set the soapaction value
```

```
        *
        * @param    string $soapaction
        * @access   public
        */
        function setSOAPAction($soapaction) {
                $this->setHeader('SOAPAction', "" . $soapaction . "");
        }


        /**
        * use http encoding
        *
        * @param    string $enc encoding style. supported values: gzip, deflate, or
both
        * @access   public
        */
        function setEncoding($enc='gzip, deflate') {
                if (function_exists('gzdeflate')) {
                        $this->protocol_version = '1.1';
                        $this->setHeader('Accept-Encoding', $enc);
                        if (!isset($this->outgoing_headers['Connection'])) {
```

```
                $this->setHeader('Connection', 'close');
                $this->persistentConnection = false;
            }
            // deprecated as of PHP 5.3.0
            //set_magic_quotes_runtime(0);
            $this->encoding = $enc;
        }
}

/**
* set proxy info here
*
* @param    string $proxyhost use an empty string to remove proxy
* @param    string $proxyport
* @param    string $proxyusername
* @param    string $proxypassword
* @param    string $proxyauthtype (basic|ntlm)
* @access   public
*/
```

```
        function setProxy($proxyhost, $proxyport, $proxyusername = '',
$proxypassword = '', $proxyauthtype = 'basic') {
                if ($proxyhost) {
                        $this->proxy = array(
                                'host' => $proxyhost,
                                'port' => $proxyport,
                                'username' => $proxyusername,
                                'password' => $proxypassword,
                                'authtype' => $proxyauthtype
                        );
                        if ($proxyusername != '' && $proxypassword != '' &&
$proxyauthtype = 'basic') {
                                $this->setHeader('Proxy-Authorization', ' Basic
'.base64_encode($proxyusername.':'.$proxypassword));
                        }
                } else {
                        $this->debug('remove proxy');
                        $proxy = null;
                        unsetHeader('Proxy-Authorization');
                }
        }
```

```
        /**
         * Test if the given string starts with a header that is to be skipped.
         * Skippable headers result from chunked transfer and proxy requests.
         *
         * @param      string $data The string to check.
         * @returns    boolean      Whether a skippable header was found.
         * @access     private
         */
        function isSkippableCurlHeader(&$data) {
                $skipHeaders = array(          'HTTP/1.1 100',
                                                        'HTTP/1.0 301',
                                                        'HTTP/1.1 301',
                                                        'HTTP/1.0 302',
                                                        'HTTP/1.1 302',
                                                        'HTTP/1.0 401',
                                                        'HTTP/1.1 401',
                                                        'HTTP/1.0 200 Connection
established');
```

```
        foreach ($skipHeaders as $hd) {
                $prefix = substr($data, 0, strlen($hd));
                if ($prefix == $hd) return true;
        }

        return false;
}

/**
* decode a string that is encoded w/ "chunked' transfer encoding
* as defined in RFC2068 19.4.6
*
* @param    string $buffer
* @param    string $lb
* @returns      string
* @access   public
* @deprecated
*/
function decodeChunked($buffer, $lb){
        // length := 0
```

```
$length = 0;
$new = '';

// read chunk-size, chunk-extension (if any) and CRLF
// get the position of the linebreak
$chunkend = strpos($buffer, $lb);
if ($chunkend == FALSE) {
	$this->debug('no linebreak found in decodeChunked');
	return $new;
}
$temp = substr($buffer,0,$chunkend);
$chunk_size = hexdec( trim($temp) );
$chunkstart = $chunkend + strlen($lb);
// while (chunk-size > 0) {
while ($chunk_size > 0) {
	$this->debug("chunkstart: $chunkstart chunk_size: $chunk_size");
	$chunkend = strpos( $buffer, $lb, $chunkstart + $chunk_size);

	// Just in case we got a broken connection
```

```
if ($chunkend == FALSE) {
    $chunk = substr($buffer,$chunkstart);
        // append chunk-data to entity-body
$new .= $chunk;
    $length += strlen($chunk);
    break;
}

// read chunk-data and CRLF
$chunk = substr($buffer,$chunkstart,$chunkend-$chunkstart);
// append chunk-data to entity-body
$new .= $chunk;
// length := length + chunk-size
$length += strlen($chunk);
// read chunk-size and CRLF
$chunkstart = $chunkend + strlen($lb);

$chunkend = strpos($buffer, $lb, $chunkstart) + strlen($lb);
if ($chunkend == FALSE) {
        break; //Just in case we got a broken connection
```

```
                }

                $temp = substr($buffer,$chunkstart,$chunkend-$chunkstart);

                $chunk_size = hexdec( trim($temp) );

                $chunkstart = $chunkend;

        }

        return $new;

}


/**
 * Writes the payload, including HTTP headers, to $this->outgoing_payload.
 *
 * @param      string $data HTTP body
 * @param      string $cookie_str data for HTTP Cookie header
 * @return     void
 * @access     private
 */
function buildPayload($data, $cookie_str = '') {
        // Note: for cURL connections, $this->outgoing_payload is ignored,
        // as is the Content-Length header, but these are still created as
        // debugging guides.
```

```
// add content-length header
if ($this->request_method != 'GET') {
        $this->setHeader('Content-Length', strlen($data));
}

// start building outgoing payload:
if ($this->proxy) {
        $uri = $this->url;
} else {
        $uri = $this->uri;
}
$req = "$this->request_method $uri HTTP/$this->protocol_version";
$this->debug("HTTP request: $req");
$this->outgoing_payload = "$req\r\n";

// loop thru headers, serializing
foreach($this->outgoing_headers as $k => $v){
        $hdr = $k.': '.$v;
        $this->debug("HTTP header: $hdr");
```

```
                $this->outgoing_payload .= "$hdr\r\n";
        }

        // add any cookies
        if ($cookie_str != '') {
                $hdr = 'Cookie: '.$cookie_str;
                $this->debug("HTTP header: $hdr");
                $this->outgoing_payload .= "$hdr\r\n";
        }

        // header/body separator
        $this->outgoing_payload .= "\r\n";

        // add data
        $this->outgoing_payload .= $data;
}

/**
* sends the SOAP request via HTTP[S]
*
```

```
        * @param    string $data message data
        * @param       array $cookies cookies to send
        * @return       boolean          true if OK, false if problem
        * @access   private
        */
        function sendRequest($data, $cookies = NULL) {
                // build cookie string
                $cookie_str = $this->getCookiesForRequest($cookies, (($this->scheme
== 'ssl') || ($this->scheme == 'https')));


                // build payload
                $this->buildPayload($data, $cookie_str);


          if ($this->io_method() == 'socket') {
                // send payload
                if(!fputs($this->fp, $this->outgoing_payload, strlen($this-
>outgoing_payload))) {
                        $this->setError('couldn\'t write message data to socket');
                        $this->debug('couldn\'t write message data to socket');
                        return false;
                }
```

```
                $this->debug('wrote data to socket, length = ' . strlen($this->outgoing_payload));
                return true;
            } else if ($this->io_method() == 'curl') {
                // set payload
                // cURL does say this should only be the verb, and in fact it
                // turns out that the URI and HTTP version are appended to this, which
                // some servers refuse to work with (so we no longer use this method!)
                //$this->setCurlOption(CURLOPT_CUSTOMREQUEST, $this->outgoing_payload);
                $curl_headers = array();
                foreach($this->outgoing_headers as $k => $v){
                    if ($k == 'Connection' || $k == 'Content-Length' || $k == 'Host' || $k == 'Authorization' || $k == 'Proxy-Authorization') {
                        $this->debug("Skip cURL header $k: $v");
                    } else {
                        $curl_headers[] = "$k: $v";
                    }
                }
                if ($cookie_str != '') {
                    $curl_headers[] = 'Cookie: ' . $cookie_str;
```

```
            }
            $this->setCurlOption(CURLOPT_HTTPHEADER, $curl_headers);
            $this->debug('set cURL HTTP headers');
            if ($this->request_method == "POST") {
                    $this->setCurlOption(CURLOPT_POST, 1);
                    $this->setCurlOption(CURLOPT_POSTFIELDS, $data);
                    $this->debug('set cURL POST data');
            } else {
            }
            // insert custom user-set cURL options
            foreach ($this->ch_options as $key => $val) {
                    $this->setCurlOption($key, $val);
            }

            $this->debug('set cURL payload');
            return true;
      }
    }

    /**
```

```
        * gets the SOAP response via HTTP[S]
        *
        * @return       string the response (also sets member variables like incoming_payload)
        * @access   private
        */
        function getResponse(){
                $this->incoming_payload = '';

          if ($this->io_method() == 'socket') {
            // loop until headers have been retrieved
            $data = '';
            while (!isset($lb)){

                        // We might EOF during header read.
                        if(feof($this->fp)) {
                                $this->incoming_payload = $data;
                                $this->debug('found no headers before EOF after length ' . strlen($data));
                                $this->debug("received before EOF:\n" . $data);
                                $this->setError('server failed to send headers');
```

```
                return false;
            }

            $tmp = fgets($this->fp, 256);
            $tmplen = strlen($tmp);
            $this->debug("read line of $tmplen bytes: " . trim($tmp));

            if ($tmplen == 0) {
                $this->incoming_payload = $data;
                $this->debug('socket read of headers timed out after length ' . strlen($data));
                $this->debug("read before timeout: " . $data);
                $this->setError('socket read of headers timed out');
                return false;
            }

            $data .= $tmp;
            $pos = strpos($data,"\r\n\r\n");
            if($pos > 1){
                $lb = "\r\n";
```

```
        } else {
                $pos = strpos($data,"\n\n");
                if($pos > 1){
                        $lb = "\n";
                }
        }
        // remove 100 headers
        if (isset($lb) && preg_match('/^HTTP\/1.1 100/',$data)) {
                unset($lb);
                $data = '';
        }//
}
// store header data
$this->incoming_payload .= $data;
$this->debug('found end of headers after length ' . strlen($data));
// process headers
$header_data = trim(substr($data,0,$pos));
$header_array = explode($lb,$header_data);
$this->incoming_headers = array();
$this->incoming_cookies = array();
```

```
                foreach($header_array as $header_line){
                        $arr = explode(':',$header_line, 2);
                        if(count($arr) > 1){
                                $header_name = strtolower(trim($arr[0]));
                                $this->incoming_headers[$header_name] =
trim($arr[1]);
                                if ($header_name == 'set-cookie') {
                                        // TODO: allow multiple cookies from
parseCookie
                                        $cookie = $this->parseCookie(trim($arr[1]));
                                        if ($cookie) {
                                                $this->incoming_cookies[] = $cookie;
                                                $this->debug('found cookie: ' .
$cookie['name'] . ' = ' . $cookie['value']);
                                        } else {
                                                $this->debug('did not find cookie in ' .
trim($arr[1]));
                                        }
                        }
                        } else if (isset($header_name)) {
                                // append continuation line to previous header
```

```
                                $this->incoming_headers[$header_name] .= $lb . ' ' .
$header_line;
                        }
                }

                // loop until msg has been received
                if (isset($this->incoming_headers['transfer-encoding']) &&
strtolower($this->incoming_headers['transfer-encoding']) == 'chunked') {
                        $content_length =  2147483647;       // ignore any content-
length header
                        $chunked = true;
                        $this->debug("want to read chunked content");
                } elseif (isset($this->incoming_headers['content-length'])) {
                        $content_length = $this->incoming_headers['content-length'];
                        $chunked = false;
                        $this->debug("want to read content of length
$content_length");
                } else {
                        $content_length =  2147483647;
                        $chunked = false;
                        $this->debug("want to read content to EOF");
```

```
                }
                $data = '';
                do {
                        if ($chunked) {
                                $tmp = fgets($this->fp, 256);
                                $tmplen = strlen($tmp);
                                $this->debug("read chunk line of $tmplen bytes");
                                if ($tmplen == 0) {
                                        $this->incoming_payload = $data;
                                        $this->debug('socket read of chunk length timed
out after length ' . strlen($data));
                                        $this->debug("read before timeout:\n" . $data);
                                        $this->setError('socket read of chunk length
timed out');
                                        return false;
                                }
                                $content_length = hexdec(trim($tmp));
                                $this->debug("chunk length $content_length");
                        }
                        $strlen = 0;
                    while (($strlen < $content_length) && (!feof($this->fp))) {
```

```
                        $readlen = min(8192, $content_length - $strlen);
                                $tmp = fread($this->fp, $readlen);
                                $tmplen = strlen($tmp);
                                $this->debug("read buffer of $tmplen bytes");
                                if (($tmplen == 0) && (!feof($this->fp))) {
                                        $this->incoming_payload = $data;
                                        $this->debug('socket read of body timed out
after length ' . strlen($data));
                                        $this->debug("read before timeout:\n" . $data);
                                        $this->setError('socket read of body timed out');
                                        return false;
                                }
                                $strlen += $tmplen;
                                $data .= $tmp;
                        }
                        if ($chunked && ($content_length > 0)) {
                                $tmp = fgets($this->fp, 256);
                                $tmplen = strlen($tmp);
                                $this->debug("read chunk terminator of $tmplen
bytes");
                                if ($tmplen == 0) {
```

```
                                        $this->incoming_payload = $data;
                                        $this->debug('socket read of chunk terminator
timed out after length ' . strlen($data));
                                        $this->debug("read before timeout:\n" . $data);
                                        $this->setError('socket read of chunk terminator
timed out');
                                        return false;
                                }
                        }
                } while ($chunked && ($content_length > 0) && (!feof($this->fp)));
                if (feof($this->fp)) {
                        $this->debug('read to EOF');
                }
                $this->debug('read body of length ' . strlen($data));
                $this->incoming_payload .= $data;
                $this->debug('received a total of '.strlen($this->incoming_payload).'
bytes of data from server');


                // close filepointer
                if(
```

```
                (isset($this->incoming_headers['connection']) &&
strtolower($this->incoming_headers['connection']) == 'close') ||
                (! $this->persistentConnection) || feof($this->fp)){
                fclose($this->fp);
                $this->fp = false;
                $this->debug('closed socket');
        }

        // connection was closed unexpectedly
        if($this->incoming_payload == ''){
                $this->setError('no response from server');
                return false;
        }

        // decode transfer-encoding
//      if(isset($this->incoming_headers['transfer-encoding']) &&
strtolower($this->incoming_headers['transfer-encoding']) == 'chunked'){
//              if(!$data = $this->decodeChunked($data, $lb)){
//                      $this->setError('Decoding of chunked data failed');
//                      return false;
//              }
```

```
                        //print "<pre>\nde-chunked:\n---------------\n$data\n\n----------
---\n</pre>";
                        // set decoded payload
//                      $this->incoming_payload = $header_data.$lb.$lb.$data;
//                  }

            } else if ($this->io_method() == 'curl') {
                    // send and receive
                    $this->debug('send and receive with cURL');
                    $this->incoming_payload = curl_exec($this->ch);
                    $data = $this->incoming_payload;

        $cErr = curl_error($this->ch);
                    if ($cErr != '') {
            $err = 'cURL ERROR: '.curl_errno($this->ch).': '.$cErr.'<br>';
            // TODO: there is a PHP bug that can cause this to SEGV for
CURLINFO_CONTENT_TYPE
                        foreach(curl_getinfo($this->ch) as $k => $v){
                            $err .= "$k: $v<br>";
                        }
                        $this->debug($err);
```

```
            $this->setError($err);
            curl_close($this->ch);
return false;
} else {
            //echo '<pre>';
            //var_dump(curl_getinfo($this->ch));
            //echo '</pre>';
}
// close curl
$this->debug('No cURL error, closing cURL');
curl_close($this->ch);

// try removing skippable headers
$savedata = $data;
while ($this->isSkippableCurlHeader($data)) {
            $this->debug("Found HTTP header to skip");
            if ($pos = strpos($data,"\r\n\r\n")) {
                        $data = ltrim(substr($data,$pos));
            } elseif($pos = strpos($data,"\n\n") ) {
                        $data = ltrim(substr($data,$pos));
```

```
        }
}

if ($data == '') {
        // have nothing left; just remove 100 header(s)
        $data = $savedata;
        while (preg_match('/^HTTP\/1.1 100/',$data)) {
                if ($pos = strpos($data,"\r\n\r\n")) {
                        $data = ltrim(substr($data,$pos));
                } elseif($pos = strpos($data,"\n\n") ) {
                        $data = ltrim(substr($data,$pos));
                }
        }
}

// separate content from HTTP headers
if ($pos = strpos($data,"\r\n\r\n")) {
        $lb = "\r\n";
} elseif( $pos = strpos($data,"\n\n")) {
        $lb = "\n";
```

```
                } else {
                        $this->debug('no proper separation of headers and document');
                        $this->setError('no proper separation of headers and
document');
                        return false;
                }
                $header_data = trim(substr($data,0,$pos));
                $header_array = explode($lb,$header_data);
                $data = ltrim(substr($data,$pos));
                $this->debug('found proper separation of headers and document');
                $this->debug('cleaned data, stringlen: '.strlen($data));
                // clean headers
                foreach ($header_array as $header_line) {
                        $arr = explode(':',$header_line,2);
                        if(count($arr) > 1){
                                $header_name = strtolower(trim($arr[0]));
                                $this->incoming_headers[$header_name] =
trim($arr[1]);
                                if ($header_name == 'set-cookie') {
                                        // TODO: allow multiple cookies from
parseCookie
```

```
                                        $cookie = $this->parseCookie(trim($arr[1]));
                                        if ($cookie) {
                                                $this->incoming_cookies[] = $cookie;
                                                $this->debug('found cookie: ' .
$cookie['name'] . ' = ' . $cookie['value']);
                                        } else {
                                                $this->debug('did not find cookie in ' .
trim($arr[1]));
                                        }
                                }
                                } else if (isset($header_name)) {
                                        // append continuation line to previous header
                                        $this->incoming_headers[$header_name] .= $lb . ' ' .
$header_line;
                                }
                        }
                }

                $this->response_status_line = $header_array[0];
                $arr = explode(' ', $this->response_status_line, 3);
                $http_version = $arr[0];
```

```
                $http_status = intval($arr[1]);
                $http_reason = count($arr) > 2 ? $arr[2] : '';

                // see if we need to resend the request with http digest authentication
                if (isset($this->incoming_headers['location']) && ($http_status == 301 || $http_status == 302)) {
                        $this->debug("Got $http_status $http_reason with Location: " . $this->incoming_headers['location']);
                        $this->setURL($this->incoming_headers['location']);
                        $this->tryagain = true;
                        return false;
                }

                // see if we need to resend the request with http digest authentication
                if (isset($this->incoming_headers['www-authenticate']) && $http_status == 401) {
                        $this->debug("Got 401 $http_reason with WWW-Authenticate: " . $this->incoming_headers['www-authenticate']);
                        if (strstr($this->incoming_headers['www-authenticate'], "Digest ")) {
                                $this->debug('Server wants digest authentication');
                                // remove "Digest " from our elements
```

```
                    $digestString = str_replace('Digest ', '', $this->incoming_headers['www-authenticate']);

                    // parse elements into array
                    $digestElements = explode(',', $digestString);
                    foreach ($digestElements as $val) {
                        $tempElement = explode('=', trim($val), 2);
                        $digestRequest[$tempElement[0]] = str_replace("\"", '', $tempElement[1]);
                    }

                    // should have (at least) qop, realm, nonce
                    if (isset($digestRequest['nonce'])) {
                        $this->setCredentials($this->username, $this->password, 'digest', $digestRequest);
                        $this->tryagain = true;
                        return false;
                    }
                }
                $this->debug('HTTP authentication failed');
                $this->setError('HTTP authentication failed');
```

```
                return false;
            }
            if (
                ($http_status >= 300 && $http_status <= 307) ||
                ($http_status >= 400 && $http_status <= 417) ||
                ($http_status >= 501 && $http_status <= 505)
              ) {
                $this->setError("Unsupported HTTP response status $http_status $http_reason (soapclient->response has contents of the response)");
                return false;
            }
            // decode content-encoding
            if(isset($this->incoming_headers['content-encoding']) && $this->incoming_headers['content-encoding'] != ''){
                if(strtolower($this->incoming_headers['content-encoding']) == 'deflate' || strtolower($this->incoming_headers['content-encoding']) == 'gzip'){
                // if decoding works, use it. else assume data wasn't gzencoded
                if(function_exists('gzinflate')){
                        //$timer->setMarker('starting decoding of gzip/deflated content');
                        // IIS 5 requires gzinflate instead of gzuncompress (similar to IE 5 and gzdeflate v. gzcompress)
```

```
                                        // this means there are no Zlib headers, although
there should be
                                        $this->debug('The gzinflate function exists');
                                        $datalen = strlen($data);
                                        if ($this->incoming_headers['content-encoding']
== 'deflate') {
                                                if ($degzdata = @gzinflate($data)) {
                                                $data = $degzdata;
                                                $this->debug('The payload has been
inflated to ' . strlen($data) . ' bytes');
                                                if (strlen($data) < $datalen) {
                                                        // test for the case that the payload
has been compressed twice
                                                        $this->debug('The inflated
payload is smaller than the gzipped one; try again');
                                                                if ($degzdata =
@gzinflate($data)) {
                                                                $data = $degzdata;
                                                                $this->debug('The payload
has been inflated again to ' . strlen($data) . ' bytes');
                                                                }
                                                }
```

```
                } else {
                    $this->debug('Error using gzinflate to
inflate the payload');
                    $this->setError('Error using gzinflate to
inflate the payload');
                }
                } elseif ($this->incoming_headers['content-
encoding'] == 'gzip') {
                    if ($degzdata = @gzinflate(substr($data,
10))) { // do our best
                        $data = $degzdata;
                    $this->debug('The payload has been un-
gzipped to ' . strlen($data) . ' bytes');
                    if (strlen($data) < $datalen) {
                        // test for the case that the payload
has been compressed twice
                        $this->debug('The un-gzipped
payload is smaller than the gzipped one; try again');
                            if ($degzdata =
@gzinflate(substr($data, 10))) {
                            $data = $degzdata;
                            $this->debug('The payload
has been un-gzipped again to ' . strlen($data) . ' bytes');
```

```
                                        }
                                    }
                                } else {
                                    $this->debug('Error using gzinflate to un-
gzip the payload');
                                        $this->setError('Error using
gzinflate to un-gzip the payload');
                                }
                                }
                                //$timer->setMarker('finished decoding of
gzip/deflated content');
                                //print "<xmp>\nde-inflated:\n----------
\n$data\n----------\n</xmp>";
                                // set decoded payload
                                $this->incoming_payload =
$header_data.$lb.$lb.$data;
                            } else {
                                $this->debug('The server sent compressed data.
Your php install must have the Zlib extension compiled in to support this.');
                                $this->setError('The server sent compressed
data. Your php install must have the Zlib extension compiled in to support this.');
                            }
```

```
                } else {
                        $this->debug('Unsupported Content-Encoding ' . $this->incoming_headers['content-encoding']);
                        $this->setError('Unsupported Content-Encoding ' . $this->incoming_headers['content-encoding']);
                }
        } else {
                $this->debug('No Content-Encoding header');
        }

        if(strlen($data) == 0){
                $this->debug('no data after headers!');
                $this->setError('no data present after HTTP headers');
                return false;
        }

        return $data;
    }

    /**
     * sets the content-type for the SOAP message to be sent
```

```
     *
     * @param     string $type the content type, MIME style
     * @param     mixed $charset character set used for encoding (or false)
     * @access    public
     */
    function setContentType($type, $charset = false) {
        $this->setHeader('Content-Type', $type . ($charset ? '; charset=' .
$charset : ''));
    }
    /**
     * specifies that an HTTP persistent connection should be used
     *
     * @return    boolean whether the request was honored by this method.
     * @access    public
     */
    function usePersistentConnection(){
        if (isset($this->outgoing_headers['Accept-Encoding'])) {
            return false;
        }
        $this->protocol_version = '1.1';
```

```
        $this->persistentConnection = true;
        $this->setHeader('Connection', 'Keep-Alive');
        return true;
}

/**
 * parse an incoming Cookie into it's parts
 *
 * @param      string $cookie_str content of cookie
 * @return     array with data of that cookie
 * @access     private
 */
/*
 * TODO: allow a Set-Cookie string to be parsed into multiple cookies
 */
function parseCookie($cookie_str) {
        $cookie_str = str_replace('; ', ';', $cookie_str) . ';';
        $data = preg_split('/;/', $cookie_str);
        $value_str = $data[0];
```

```
$cookie_param = 'domain=';
$start = strpos($cookie_str, $cookie_param);
if ($start > 0) {
        $domain = substr($cookie_str, $start + strlen($cookie_param));
        $domain = substr($domain, 0, strpos($domain, ';'));
} else {
        $domain = '';
}

$cookie_param = 'expires=';
$start = strpos($cookie_str, $cookie_param);
if ($start > 0) {
        $expires = substr($cookie_str, $start + strlen($cookie_param));
        $expires = substr($expires, 0, strpos($expires, ';'));
} else {
        $expires = '';
}

$cookie_param = 'path=';
$start = strpos($cookie_str, $cookie_param);
```

```
if ( $start > 0 ) {
        $path = substr($cookie_str, $start + strlen($cookie_param));
        $path = substr($path, 0, strpos($path, ';'));
} else {
        $path = '/';
}

$cookie_param = ';secure;';
if (strpos($cookie_str, $cookie_param) !== FALSE) {
        $secure = true;
} else {
        $secure = false;
}

$sep_pos = strpos($value_str, '=');

if ($sep_pos) {
        $name = substr($value_str, 0, $sep_pos);
        $value = substr($value_str, $sep_pos + 1);
        $cookie= array(         'name' => $name,
```

```
                    'value' => $value,
                                        'domain' => $domain,
                                        'path' => $path,
                                        'expires' => $expires,
                                        'secure' => $secure
                                        );
                return $cookie;
        }
        return false;
}

/**
 * sort out cookies for the current request
 *
 * @param      array $cookies array with all cookies
 * @param      boolean $secure is the send-content secure or not?
 * @return     string for Cookie-HTTP-Header
 * @access     private
 */
function getCookiesForRequest($cookies, $secure=false) {
```

```
                $cookie_str = '';
                if ((! is_null($cookies)) && (is_array($cookies))) {
                        foreach ($cookies as $cookie) {
                                if (! is_array($cookie)) {
                                        continue;
                                }
                                $this->debug("check cookie for validity:
".$cookie['name'].'='.$cookie['value']);
                                if ((isset($cookie['expires'])) && (!
empty($cookie['expires']))) {
                                        if (strtotime($cookie['expires']) <= time()) {
                                                $this->debug('cookie has expired');
                                                continue;
                                        }
                                }
                                if ((isset($cookie['domain'])) && (!
empty($cookie['domain']))) {
                                        $domain = preg_quote($cookie['domain']);
                                        if (! preg_match("'.*$domain$'i", $this->host)) {
                                                $this->debug('cookie has different
domain');
```

```
                                continue;
                            }
                        }
                        if ((isset($cookie['path'])) && (!
empty($cookie['path']))) {
                            $path = preg_quote($cookie['path']);
                            if (! preg_match("'^$path.*'i", $this->path)) {
                                $this->debug('cookie is for a different
path');
                                continue;
                            }
                        }
                        if ((! $secure) && (isset($cookie['secure'])) &&
($cookie['secure'])) {
                            $this->debug('cookie is secure, transport is not');
                            continue;
                        }
                        $cookie_str .= $cookie['name'] . '=' . $cookie['value'] . ';
';
                    $this->debug('add cookie to Cookie-String: ' . $cookie['name'] .
'=' . $cookie['value']);
                    }
```

```
                }
                return $cookie_str;
  }
}


?><?php




/**
*
* nusoap_server allows the user to create a SOAP server
* that is capable of receiving messages and returning responses
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class nusoap_server extends nusoap_base {
```

```
        /**
         * HTTP headers of request
         * @var array
         * @access private
         */
        var $headers = array();
        /**
         * HTTP request
         * @var string
         * @access private
         */
        var $request = '';
        /**
         * SOAP headers from request (incomplete namespace resolution; special characters not escaped) (text)
         * @var string
         * @access public
         */
        var $requestHeaders = '';
        /**
```

```
     * SOAP Headers from request (parsed)
     * @var mixed
     * @access public
     */
    var $requestHeader = NULL;
    /**
     * SOAP body request portion (incomplete namespace resolution; special characters not escaped) (text)
     * @var string
     * @access public
     */
    var $document = '';
    /**
     * SOAP payload for request (text)
     * @var string
     * @access public
     */
    var $requestSOAP = '';
    /**
     * requested method namespace URI
```

```
 * @var string
 * @access private
 */
var $methodURI = '';
/**
 * name of method requested
 * @var string
 * @access private
 */
var $methodname = '';
/**
 * method parameters from request
 * @var array
 * @access private
 */
var $methodparams = array();
/**
 * SOAP Action from request
 * @var string
 * @access private
```

```
     */
    var $SOAPAction = '';
    /**
     * character set encoding of incoming (request) messages
     * @var string
     * @access public
     */
    var $xml_encoding = '';
    /**
     * toggles whether the parser decodes element content w/ utf8_decode()
     * @var boolean
     * @access public
     */
var $decode_utf8 = true;

    /**
     * HTTP headers of response
     * @var array
     * @access public
     */
```

```
var $outgoing_headers = array();
/**
 * HTTP response
 * @var string
 * @access private
 */
var $response = '';
/**
 * SOAP headers for response (text or array of soapval or associative array)
 * @var mixed
 * @access public
 */
var $responseHeaders = '';
/**
 * SOAP payload for response (text)
 * @var string
 * @access private
 */
var $responseSOAP = '';
/**
```

```
     * method return value to place in response
     * @var mixed
     * @access private
     */
    var $methodreturn = false;
    /**
     * whether $methodreturn is a string of literal XML
     * @var boolean
     * @access public
     */
    var $methodreturnisliteralxml = false;
    /**
     * SOAP fault for response (or false)
     * @var mixed
     * @access private
     */
    var $fault = false;
    /**
     * text indication of result (for debugging)
     * @var string
```

```
     * @access private
     */
    var $result = 'successful';

    /**
     * assoc array of operations => opData; operations are added by the register()
     * method or by parsing an external WSDL definition
     * @var array
     * @access private
     */
    var $operations = array();
    /**
     * wsdl instance (if one)
     * @var mixed
     * @access private
     */
    var $wsdl = false;
    /**
     * URL for WSDL (if one)
     * @var mixed
```

```
     * @access private
     */
    var $externalWSDLURL = false;
    /**
     * whether to append debug to response as XML comment
     * @var boolean
     * @access public
     */
    var $debug_flag = false;



    /**
    * constructor
   * the optional parameter is a path to a WSDL file that you'd like to bind the server instance to.
    *
   * @param mixed $wsdl file path or URL (string), or wsdl instance (object)
    * @access   public
    */
    function nusoap_server($wsdl=false){
```

```
        parent::nusoap_base();
        // turn on debugging?
        global $debug;
        global $HTTP_SERVER_VARS;

        if (isset($_SERVER)) {
            $this->debug("_SERVER is defined:");
            $this->appendDebug($this->varDump($_SERVER));
        } elseif (isset($HTTP_SERVER_VARS)) {
            $this->debug("HTTP_SERVER_VARS is defined:");
            $this->appendDebug($this->varDump($HTTP_SERVER_VARS));
        } else {
            $this->debug("Neither _SERVER nor HTTP_SERVER_VARS is defined.");
        }

        if (isset($debug)) {
            $this->debug("In nusoap_server, set debug_flag=$debug based on global flag");
            $this->debug_flag = $debug;
```

```
        } elseif (isset($_SERVER['QUERY_STRING'])) {
            $qs = explode('&', $_SERVER['QUERY_STRING']);
            foreach ($qs as $v) {
                if (substr($v, 0, 6) == 'debug=') {
                    $this->debug("In nusoap_server, set debug_flag=" . substr($v, 6) . " based on query string #1");
                    $this->debug_flag = substr($v, 6);
                }
            }
        } elseif (isset($HTTP_SERVER_VARS['QUERY_STRING'])) {
            $qs = explode('&', $HTTP_SERVER_VARS['QUERY_STRING']);
            foreach ($qs as $v) {
                if (substr($v, 0, 6) == 'debug=') {
                    $this->debug("In nusoap_server, set debug_flag=" . substr($v, 6) . " based on query string #2");
                    $this->debug_flag = substr($v, 6);
                }
            }
        }
```

```
        // wsdl
        if($wsdl){
            $this->debug("In nusoap_server, WSDL is specified");
            if (is_object($wsdl) && (get_class($wsdl) == 'wsdl')) {
                $this->wsdl = $wsdl;
                $this->externalWSDLURL = $this->wsdl->wsdl;
                $this->debug('Use existing wsdl instance from ' . $this->externalWSDLURL);
            } else {
                $this->debug('Create wsdl from ' . $wsdl);
                $this->wsdl = new wsdl($wsdl);
                $this->externalWSDLURL = $wsdl;
            }
            $this->appendDebug($this->wsdl->getDebug());
            $this->wsdl->clearDebug();
            if($err = $this->wsdl->getError()){
                die('WSDL ERROR: '.$err);
            }
        }
    }
```

```
/**
* processes request and returns response
*
* @param    string $data usually is the value of $HTTP_RAW_POST_DATA
* @access   public
*/
function service($data){
        global $HTTP_SERVER_VARS;

        if (isset($_SERVER['REQUEST_METHOD'])) {
                $rm = $_SERVER['REQUEST_METHOD'];
        } elseif (isset($HTTP_SERVER_VARS['REQUEST_METHOD'])) {
                $rm = $HTTP_SERVER_VARS['REQUEST_METHOD'];
        } else {
                $rm = '';
        }

        if (isset($_SERVER['QUERY_STRING'])) {
                $qs = $_SERVER['QUERY_STRING'];
```

```
        } elseif (isset($HTTP_SERVER_VARS['QUERY_STRING'])) {
                $qs = $HTTP_SERVER_VARS['QUERY_STRING'];
        } else {
                $qs = '';
        }
        $this->debug("In service, request method=$rm query string=$qs
strlen(\$data)=" . strlen($data));

        if ($rm == 'POST') {
                $this->debug("In service, invoke the request");
                $this->parse_request($data);
                if (! $this->fault) {
                        $this->invoke_method();
                }
                if (! $this->fault) {
                        $this->serialize_return();
                }
                $this->send_response();
        } elseif (preg_match('/wsdl/', $qs) ){
                $this->debug("In service, this is a request for WSDL");
```

```
                    if ($this->externalWSDLURL){
            if (strpos($this->externalWSDLURL, "http://") !== false) { // assume URL
                        $this->debug("In service, re-direct for WSDL");
                        header('Location: '.$this->externalWSDLURL);
            } else { // assume file
                        $this->debug("In service, use file passthru for WSDL");
             header("Content-Type: text/xml\r\n");
                        $pos = strpos($this->externalWSDLURL, "file://");
                        if ($pos === false) {
                                $filename = $this->externalWSDLURL;
                        } else {
                                $filename = substr($this->externalWSDLURL,
$pos + 7);
                        }
             $fp = fopen($this->externalWSDLURL, 'r');
             fpassthru($fp);
            }
                    } elseif ($this->wsdl) {
                        $this->debug("In service, serialize WSDL");
                        header("Content-Type: text/xml; charset=ISO-8859-
1\r\n");
```

```
                print $this->wsdl->serialize($this->debug_flag);
                if ($this->debug_flag) {
                    $this->debug('wsdl:');
                    $this->appendDebug($this->varDump($this->wsdl));
                    print $this->getDebugAsXMLComment();
                }
            } else {
                $this->debug("In service, there is no WSDL");
                header("Content-Type: text/html; charset=ISO-8859-1\r\n");
                print "This service does not provide WSDL";
            }
        } elseif ($this->wsdl) {
            $this->debug("In service, return Web description");
            print $this->wsdl->webDescription();
        } else {
            $this->debug("In service, no Web description");
            header("Content-Type: text/html; charset=ISO-8859-1\r\n");
            print "This service does not provide a Web description";
        }
```

```
}


/**
* parses HTTP request headers.
*
* The following fields are set by this function (when successful)
*
* headers
* request
* xml_encoding
* SOAPAction
*
* @access   private
*/
function parse_http_headers() {
        global $HTTP_SERVER_VARS;


        $this->request = '';
        $this->SOAPAction = '';
        if(function_exists('getallheaders')){
```

```
                $this->debug("In parse_http_headers, use getallheaders");
                $headers = getallheaders();
                foreach($headers as $k=>$v){
                        $k = strtolower($k);
                        $this->headers[$k] = $v;
                        $this->request .= "$k: $v\r\n";
                        $this->debug("$k: $v");
                }
                // get SOAPAction header
                if(isset($this->headers['soapaction'])){
                        $this->SOAPAction = str_replace("","",$this-
>headers['soapaction']);
                }
                // get the character encoding of the incoming request
                if(isset($this->headers['content-type']) && strpos($this-
>headers['content-type'],'=')){
                        $enc = str_replace("","",substr(strstr($this-
>headers["content-type"],'='),1));
                        if(preg_match('/^(ISO-8859-1|US-ASCII|UTF-
8)$/i',$enc)){
                                $this->xml_encoding = strtoupper($enc);
```

```
                } else {
                        $this->xml_encoding = 'US-ASCII';
                }
        } else {
                // should be US-ASCII for HTTP 1.0 or ISO-8859-1 for
HTTP 1.1
                $this->xml_encoding = 'ISO-8859-1';
        }
    } elseif(isset($_SERVER) && is_array($_SERVER)){
        $this->debug("In parse_http_headers, use _SERVER");
        foreach ($_SERVER as $k => $v) {
                if (substr($k, 0, 5) == 'HTTP_') {
                        $k = str_replace(' ', '-', strtolower(str_replace('_', '
', substr($k, 5))));
                } else {
                        $k = str_replace(' ', '-', strtolower(str_replace('_', '
', $k)));
                }
                if ($k == 'soapaction') {
                        // get SOAPAction header
                        $k = 'SOAPAction';
```

```
                    $v = str_replace("'", '', $v);
                    $v = str_replace('\\', '', $v);
                    $this->SOAPAction = $v;
                } else if ($k == 'content-type') {
                    // get the character encoding of the incoming
request
                    if (strpos($v, '=')) {
                        $enc = substr(strstr($v, '='), 1);
                        $enc = str_replace("'", '', $enc);
                        $enc = str_replace('\\', '', $enc);
                        if (preg_match('/^(ISO-8859-1|US-
ASCII|UTF-8)$/i',$enc)) {
                            $this->xml_encoding =
strtoupper($enc);
                        } else {
                            $this->xml_encoding = 'US-
ASCII';
                        }
                    } else {
                        // should be US-ASCII for HTTP 1.0 or
ISO-8859-1 for HTTP 1.1
                        $this->xml_encoding = 'ISO-8859-1';
```

```
                    }
                }
                $this->headers[$k] = $v;
                $this->request .= "$k: $v\r\n";
                $this->debug("$k: $v");
            }
        } elseif (is_array($HTTP_SERVER_VARS)) {
            $this->debug("In parse_http_headers, use HTTP_SERVER_VARS");
            foreach ($HTTP_SERVER_VARS as $k => $v) {
                if (substr($k, 0, 5) == 'HTTP_') {
                    $k = str_replace(' ', '-', strtolower(str_replace('_', ' ', substr($k, 5))));
                    $k = strtolower(substr($k, 5));
                } else {
                    $k = str_replace(' ', '-', strtolower(str_replace('_', ' ', $k)));
                    $k = strtolower($k);
                }
                if ($k == 'soapaction') {
                    // get SOAPAction header
                    $k = 'SOAPAction';
                    $v = str_replace("", '', $v);
```

```
                    $v = str_replace('\\', '', $v);
                    $this->SOAPAction = $v;
                } else if ($k == 'content-type') {
                    // get the character encoding of the incoming
request
                    if (strpos($v, '=')) {
                        $enc = substr(strstr($v, '='), 1);
                        $enc = str_replace('"', '', $enc);
                        $enc = str_replace('\\', '', $enc);
                        if (preg_match('/^(ISO-8859-1|US-
ASCII|UTF-8)$/i',$enc)) {
                            $this->xml_encoding =
strtoupper($enc);
                        } else {
                            $this->xml_encoding = 'US-
ASCII';
                        }
                    } else {
                        // should be US-ASCII for HTTP 1.0 or
ISO-8859-1 for HTTP 1.1
                        $this->xml_encoding = 'ISO-8859-1';
                    }
```

```
                        }
                        $this->headers[$k] = $v;
                        $this->request .= "$k: $v\r\n";
                        $this->debug("$k: $v");
                }
        } else {
                $this->debug("In parse_http_headers, HTTP headers not accessible");
                $this->setError("HTTP headers not accessible");
        }
}

/**
* parses a request
*
* The following fields are set by this function (when successful)
*
* headers
* request
* xml_encoding
```

```
* SOAPAction
* request
* requestSOAP
* methodURI
* methodname
* methodparams
* requestHeaders
* document
*
* This sets the fault field on error
*
* @param    string $data XML string
* @access   private
*/
function parse_request($data='') {
        $this->debug('entering parse_request()');
        $this->parse_http_headers();
        $this->debug('got character encoding: '.$this->xml_encoding);
        // uncompress if necessary
```

```
                if (isset($this->headers['content-encoding']) && $this-
>headers['content-encoding'] != '') {
                    $this->debug('got content encoding: ' . $this->headers['content-
encoding']);
                    if ($this->headers['content-encoding'] == 'deflate' || $this-
>headers['content-encoding'] == 'gzip') {
                    // if decoding works, use it. else assume data wasn't gzencoded
                        if (function_exists('gzuncompress')) {
                            if ($this->headers['content-encoding'] ==
'deflate' && $degzdata = @gzuncompress($data)) {
                                $data = $degzdata;
                            } elseif ($this->headers['content-encoding'] ==
'gzip' && $degzdata = gzinflate(substr($data, 10))) {
                                $data = $degzdata;
                            } else {
                                $this->fault('SOAP-ENV:Client', 'Errors
occurred when trying to decode the data');
                                return;
                            }
                        } else {
                            $this->fault('SOAP-ENV:Client', 'This Server
does not support compressed data');
```

```
                                return;
                            }
                        }
                }
                $this->request .= "\r\n".$data;
                $data = $this->parseRequest($this->headers, $data);
                $this->requestSOAP = $data;
                $this->debug('leaving parse_request');
}

/**
* invokes a PHP function for the requested SOAP method
*
* The following fields are set by this function (when successful)
*
* methodreturn
*
* Note that the PHP function that is called may also set the following
* fields to affect the response sent to the client
*
```

```
        * responseHeaders
        * outgoing_headers
        *
        * This sets the fault field on error
        *
        * @access   private
        */
        function invoke_method() {
                $this->debug('in invoke_method, methodname=' . $this->methodname
. ' methodURI=' . $this->methodURI . ' SOAPAction=' . $this->SOAPAction);


                //
                // if you are debugging in this area of the code, your service uses a
class to implement methods,
                // you use SOAP RPC, and the client is .NET, please be aware of the
following...
                // when the .NET wsdl.exe utility generates a proxy, it will remove the
'.' or '..' from the
                // method name.  that is fine for naming the .NET methods.  it is not
fine for properly constructing
                // the XML request and reading the XML response.  you need to add
the RequestElementName and
```

```
                // ResponseElementName to the
System.Web.Services.Protocols.SoapRpcMethodAttribute that wsdl.exe
                // generates for the method.  these parameters are used to specify the
correct XML element names
                // for .NET to use, i.e. the names with the '.' in them.
                //
                $orig_methodname = $this->methodname;
                if ($this->wsdl) {
                        if ($this->opData = $this->wsdl->getOperationData($this-
>methodname)) {
                                $this->debug('in invoke_method, found WSDL
operation=' . $this->methodname);
                                $this->appendDebug('opData=' . $this->varDump($this-
>opData));
                        } elseif ($this->opData = $this->wsdl-
>getOperationDataForSoapAction($this->SOAPAction)) {
                                // Note: hopefully this case will only be used for doc/lit,
since rpc services should have wrapper element
                                $this->debug('in invoke_method, found WSDL
soapAction=' . $this->SOAPAction . ' for operation=' . $this->opData['name']);
                                $this->appendDebug('opData=' . $this->varDump($this-
>opData));
                                $this->methodname = $this->opData['name'];
```

```
                } else {
                        $this->debug('in invoke_method, no WSDL for operation=' . $this->methodname);
                        $this->fault('SOAP-ENV:Client', "Operation '" . $this->methodname . "' is not defined in the WSDL for this service");
                        return;
                }
        } else {
                $this->debug('in invoke_method, no WSDL to validate method');
        }

        // if a . is present in $this->methodname, we see if there is a class in scope,
        // which could be referred to. We will also distinguish between two deliminators,
        // to allow methods to be called a the class or an instance
        if (strpos($this->methodname, '..') > 0) {
                $delim = '..';
        } else if (strpos($this->methodname, '.') > 0) {
                $delim = '.';
        } else {
```

```
                $delim = '';
            }
            $this->debug("in invoke_method, delim=$delim");

            $class = '';
            $method = '';
            if (strlen($delim) > 0 && substr_count($this->methodname, $delim)
== 1) {
                $try_class = substr($this->methodname, 0, strpos($this-
>methodname, $delim));
                if (class_exists($try_class)) {
                    // get the class and method name
                    $class = $try_class;
                    $method = substr($this->methodname, strpos($this-
>methodname, $delim) + strlen($delim));
                    $this->debug("in invoke_method, class=$class
method=$method delim=$delim");
                } else {
                    $this->debug("in invoke_method, class=$try_class not
found");
                }
            } else {
```

```
                $try_class = '';
                $this->debug("in invoke_method, no class to try");
            }
            // does method exist?
            if ($class == '') {
                if (!function_exists($this->methodname)) {
                    $this->debug("in invoke_method, function '$this->methodname' not found!");
                    $this->result = 'fault: method not found';
                    $this->fault('SOAP-ENV:Client',"method '$this->methodname'('$orig_methodname') not defined in service('$try_class' '$delim')");
                    return;
                }
            } else {
                $method_to_compare = (substr(phpversion(), 0, 2) == '4.') ? strtolower($method) : $method;
                if (!in_array($method_to_compare, get_class_methods($class))) {
                    $this->debug("in invoke_method, method '$this->methodname' not found in class '$class'!");
                    $this->result = 'fault: method not found';
```

```
                        $this->fault('SOAP-ENV:Client',"method '$this-
>methodname'/'$method_to_compare'('$orig_methodname') not defined in
service/'$class'('$try_class' '$delim')");
                        return;
                }
        }

        // evaluate message, getting back parameters
        // verify that request parameters match the method's signature
        if(! $this->verify_method($this->methodname,$this-
>methodparams)){
                // debug
                $this->debug('ERROR: request not verified against method
signature');
                $this->result = 'fault: request failed validation against method
signature';
                // return fault
                $this->fault('SOAP-ENV:Client',"Operation '$this-
>methodname' not defined in service.");
                return;
        }
```

```
                // if there are parameters to pass
                $this->debug('in invoke_method, params:');
                $this->appendDebug($this->varDump($this->methodparams));
                $this->debug("in invoke_method, calling '$this->methodname'");
                if (!function_exists('call_user_func_array')) {
                        if ($class == '') {
                                $this->debug('in invoke_method, calling function using eval()');
                                $funcCall = "\$this->methodreturn = $this->methodname(";
                        } else {
                                if ($delim == '..') {
                                        $this->debug('in invoke_method, calling class method using eval()');
                                        $funcCall = "\$this->methodreturn = ".$class."::".$method."(";
                                } else {
                                        $this->debug('in invoke_method, calling instance method using eval()');
                                        // generate unique instance name
                                        $instname = "\$inst_".time();
                                        $funcCall = $instname." = new ".$class."(); ";
```

```
                                $funcCall .= "\$this->methodreturn =
".$instname."->".$method."(";
                            }
                        }
                        if ($this->methodparams) {
                            foreach ($this->methodparams as $param) {
                                if (is_array($param) || is_object($param)) {
                                    $this->fault('SOAP-ENV:Client',
'NuSOAP does not handle complexType parameters correctly when using eval;
call_user_func_array must be available');
                                    return;
                                }
                                $funcCall .= "\"$param\",";
                            }
                            $funcCall = substr($funcCall, 0, -1);
                        }
                        $funcCall .= ');';
                        $this->debug('in invoke_method, function call: '.$funcCall);
                        @eval($funcCall);
                    } else {
                        if ($class == '') {
```

```
                $this->debug('in invoke_method, calling function using
call_user_func_array()');
                $call_arg = "$this->methodname";    // straight
assignment changes $this->methodname to lower case after call_user_func_array()
            } elseif ($delim == '..') {
                $this->debug('in invoke_method, calling class method
using call_user_func_array()');
                $call_arg = array ($class, $method);
            } else {
                $this->debug('in invoke_method, calling instance
method using call_user_func_array()');
                $instance = new $class ();
                $call_arg = array(&$instance, $method);
            }
            if (is_array($this->methodparams)) {
                $this->methodreturn = call_user_func_array($call_arg,
array_values($this->methodparams));
            } else {
                $this->methodreturn = call_user_func_array($call_arg,
array());
            }
        }
```

```
        $this->debug('in invoke_method, methodreturn:');
        $this->appendDebug($this->varDump($this->methodreturn));
                $this->debug("in invoke_method, called method $this->methodname, received data of type ".gettype($this->methodreturn));
        }
        /**
        * serializes the return value from a PHP function into a full SOAP Envelope
        *
        * The following fields are set by this function (when successful)
        *
        * responseSOAP
        *
        * This sets the fault field on error
        *
        * @access   private
        */
        function serialize_return() {
                $this->debug('Entering serialize_return methodname: ' . $this->methodname . ' methodURI: ' . $this->methodURI);
                // if fault
```

```
                if (isset($this->methodreturn) && is_object($this->methodreturn) &&
((get_class($this->methodreturn) == 'soap_fault') || (get_class($this->methodreturn)
== 'nusoap_fault'))) {
                        $this->debug('got a fault object from method');
                        $this->fault = $this->methodreturn;
                        return;
                } elseif ($this->methodreturnisliteralxml) {
                        $return_val = $this->methodreturn;
                // returned value(s)
                } else {
                        $this->debug('got a(n) '.gettype($this->methodreturn).' from
method');
                        $this->debug('serializing return value');
                        if($this->wsdl){
                                if (sizeof($this->opData['output']['parts']) > 1) {
                                        $this->debug('more than one output part, so use
the method return unchanged');
                                        $opParams = $this->methodreturn;
                                } elseif (sizeof($this->opData['output']['parts']) == 1) {
                                        $this->debug('exactly one output part, so wrap
the method return in a simple array');
                                        // TODO: verify that it is not already wrapped!
```

```
                    //foreach ($this->opData['output']['parts'] as $name =>
$type) {
                        //      $this->debug('wrap in element named ' .
$name);
                    //}
                    $opParams = array($this->methodreturn);
                }
                $return_val = $this->wsdl->serializeRPCParameters($this-
>methodname,'output',$opParams);
                $this->appendDebug($this->wsdl->getDebug());
                $this->wsdl->clearDebug();
                    if($errstr = $this->wsdl->getError()){
                        $this->debug('got wsdl error: '.$errstr);
                        $this->fault('SOAP-ENV:Server', 'unable to
serialize result');
                        return;
                    }
            } else {
                    if (isset($this->methodreturn)) {
                        $return_val = $this->serialize_val($this-
>methodreturn, 'return');
                    } else {
```

```
                                        $return_val = '';
                                        $this->debug('in absence of WSDL, assume void return for backward compatibility');
                                    }
                                }
                            }
                            $this->debug('return value:');
                            $this->appendDebug($this->varDump($return_val));

                            $this->debug('serializing response');
                            if ($this->wsdl) {
                                $this->debug('have WSDL for serialization: style is ' . $this->opData['style']);
                                if ($this->opData['style'] == 'rpc') {
                                    $this->debug('style is rpc for serialization: use is ' . $this->opData['output']['use']);
                                    if ($this->opData['output']['use'] == 'literal') {
                                        // http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html R2735 says rpc/literal accessor elements should not be in a namespace
                                        if ($this->methodURI) {
```

```
                                        $payload = '<ns1:'.$this-
>methodname.'Response xmlns:ns1="'.$this-
>methodURI.'">'.$return_val.'</ns1:'.$this->methodname."Response>";
                                    } else {
                                        $payload = '<'.$this-
>methodname.'Response>'.$return_val.'</'.$this->methodname.'Response>';
                                    }
                                } else {
                                    if ($this->methodURI) {
                                        $payload = '<ns1:'.$this-
>methodname.'Response xmlns:ns1="'.$this-
>methodURI.'">'.$return_val.'</ns1:'.$this->methodname."Response>";
                                    } else {
                                        $payload = '<'.$this-
>methodname.'Response>'.$return_val.'</'.$this->methodname.'Response>';
                                    }
                                }
                            } else {
                                $this->debug('style is not rpc for serialization: assume
document');
                                $payload = $return_val;
                            }
```

```
                } else {
                        $this->debug('do not have WSDL for serialization: assume
rpc/encoded');
                        $payload = '<ns1:'.$this->methodname.'Response
xmlns:ns1="'.$this->methodURI.'">'.$return_val.'</ns1:'.$this-
>methodname."Response>";
                }
                $this->result = 'successful';
                if($this->wsdl){
                        //if($this->debug_flag){
                $this->appendDebug($this->wsdl->getDebug());
        //      }
                        if (isset($this->opData['output']['encodingStyle'])) {
                                $encodingStyle = $this-
>opData['output']['encodingStyle'];
                        } else {
                                $encodingStyle = '';
                        }
                        // Added: In case we use a WSDL, return a serialized env.
WITH the usedNamespaces.
                        $this->responseSOAP = $this-
>serializeEnvelope($payload,$this->responseHeaders,$this->wsdl-
```

```
>usedNamespaces,$this->opData['style'],$this-
>opData['output']['use'],$encodingStyle);
                } else {
                        $this->responseSOAP = $this-
>serializeEnvelope($payload,$this->responseHeaders);
                }
                $this->debug("Leaving serialize_return");
        }
        /**
        * sends an HTTP response
        *
        * The following fields are set by this function (when successful)
        *
        * outgoing_headers
        * response
        *
        * @access   private
        */
        function send_response() {
                $this->debug('Enter send_response');
                if ($this->fault) {
```

```
                $payload = $this->fault->serialize();
                $this->outgoing_headers[] = "HTTP/1.0 500 Internal Server
Error";
                $this->outgoing_headers[] = "Status: 500 Internal Server
Error";
            } else {
                $payload = $this->responseSOAP;
                // Some combinations of PHP+Web server allow the Status
                // to come through as a header.  Since OK is the default
                // just do nothing.
                // $this->outgoing_headers[] = "HTTP/1.0 200 OK";
                // $this->outgoing_headers[] = "Status: 200 OK";
            }
        // add debug data if in debug mode
            if(isset($this->debug_flag) && $this->debug_flag){
          $payload .= $this->getDebugAsXMLComment();
        }
            $this->outgoing_headers[] = "Server: $this->title Server v$this-
>version";
            preg_match('/\$Revisio' . 'n: ([^ ]+)/', $this->revision, $rev);
```

```
                $this->outgoing_headers[] = "X-SOAP-Server: $this->title/$this->version (".$rev[1].")";
                // Let the Web server decide about this
                //$this->outgoing_headers[] = "Connection: Close\r\n";
                $payload = $this->getHTTPBody($payload);
                $type = $this->getHTTPContentType();
                $charset = $this->getHTTPContentTypeCharset();
                $this->outgoing_headers[] = "Content-Type: $type" . ($charset ? '; charset=' . $charset : '');
                //begin code to compress payload - by John
                // NOTE: there is no way to know whether the Web server will also compress
                // this data.
                if (strlen($payload) > 1024 && isset($this->headers) && isset($this->headers['accept-encoding'])) {
                        if (strstr($this->headers['accept-encoding'], 'gzip')) {
                                if (function_exists('gzencode')) {
                                        if (isset($this->debug_flag) && $this->debug_flag) {
                                                $payload .= "<!-- Content being gzipped -->";
                                        }
```

```
                $this->outgoing_headers[] = "Content-Encoding:
gzip";
                $payload = gzencode($payload);
            } else {
                if (isset($this->debug_flag) && $this-
>debug_flag) {
                    $payload .= "<!-- Content will not be
gzipped: no gzencode -->";
                }
            }
        } elseif (strstr($this->headers['accept-encoding'], 'deflate')) {
            // Note: MSIE requires gzdeflate output (no Zlib header
and checksum),
            // instead of gzcompress output,
            // which conflicts with HTTP 1.1 spec
(http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.5)
            if (function_exists('gzdeflate')) {
                if (isset($this->debug_flag) && $this-
>debug_flag) {
                    $payload .= "<!-- Content being deflated
-->";
                }
```

```
                                        $this->outgoing_headers[] = "Content-Encoding:
deflate";
                                        $payload = gzdeflate($payload);
                                } else {
                                        if (isset($this->debug_flag) && $this-
>debug_flag) {
                                                $payload .= "<!-- Content will not be
deflated: no gzcompress -->";
                                        }
                                }
                        }
                }
                //end code
                $this->outgoing_headers[] = "Content-Length: ".strlen($payload);
                reset($this->outgoing_headers);
                foreach($this->outgoing_headers as $hdr){
                        header($hdr, false);
                }
                print $payload;
                $this->response = join("\r\n",$this-
>outgoing_headers)."\r\n\r\n".$payload;
```

```
}
/**
* takes the value that was created by parsing the request
* and compares to the method's signature, if available.
*
* @param      string   $operation      The operation to be invoked
* @param      array    $request        The array of parameter values
* @return     boolean          Whether the operation was found
* @access   private
*/
function verify_method($operation,$request){
        if(isset($this->wsdl) && is_object($this->wsdl)){
                if($this->wsdl->getOperationData($operation)){
                        return true;
                }
   } elseif(isset($this->operations[$operation])){
                return true;
        }
        return false;
}
```

```
        /**
        * processes SOAP message received from client
        *
        * @param      array   $headers        The HTTP headers
        * @param      string  $data           unprocessed request data from client
        * @return     mixed   value of the message, decoded into a PHP type
        * @access   private
        */
    function parseRequest($headers, $data) {
                $this->debug('Entering parseRequest() for data of length ' .
strlen($data) . ' headers:');
                $this->appendDebug($this->varDump($headers));
        if (!isset($headers['content-type'])) {
                        $this->setError('Request not of type text/xml (no content-type
header)');
                        return false;
        }
                if (!strstr($headers['content-type'], 'text/xml')) {
                        $this->setError('Request not of type text/xml');
                        return false;
                }
```

```
                if (strpos($headers['content-type'], '=')) {
                        $enc = str_replace("", '', substr(strstr($headers["content-type"],
'='), 1));
                        $this->debug('Got response encoding: ' . $enc);
                        if(preg_match('/^(ISO-8859-1|US-ASCII|UTF-8)$/i',$enc)){
                                $this->xml_encoding = strtoupper($enc);
                        } else {
                                $this->xml_encoding = 'US-ASCII';
                        }
                } else {
                        // should be US-ASCII for HTTP 1.0 or ISO-8859-1 for HTTP
1.1
                        $this->xml_encoding = 'ISO-8859-1';
                }
                $this->debug('Use encoding: ' . $this->xml_encoding . ' when creating
nusoap_parser');
                // parse response, get soap parser obj
                $parser = new nusoap_parser($data,$this->xml_encoding,'',$this-
>decode_utf8);
                // parser debug
                $this->debug("parser debug: \n".$parser->getDebug());
```

```
                // if fault occurred during message parsing
                if($err = $parser->getError()){
                        $this->result = 'fault: error in msg parsing: '.$err;
                        $this->fault('SOAP-ENV:Client',"error in msg
parsing:\n".$err);
                // else successfully parsed request into soapval object
                } else {
                        // get/set methodname
                        $this->methodURI = $parser->root_struct_namespace;
                        $this->methodname = $parser->root_struct_name;
                        $this->debug('methodname: '.$this->methodname.' methodURI:
'.$this->methodURI);
                        $this->debug('calling parser->get_soapbody()');
                        $this->methodparams = $parser->get_soapbody();
                        // get SOAP headers
                        $this->requestHeaders = $parser->getHeaders();
                        // get SOAP Header
                        $this->requestHeader = $parser->get_soapheader();
        // add document for doclit support
        $this->document = $parser->document;
                }
```

```
 }

/**

* gets the HTTP body for the current response.

*

* @param string $soapmsg The SOAP payload

* @return string The HTTP body, which includes the SOAP payload

* @access private

*/

function getHTTPBody($soapmsg) {

        return $soapmsg;

}

/**

* gets the HTTP content type for the current response.

*

* Note: getHTTPBody must be called before this.

*

* @return string the HTTP content type for the current response.

* @access private

*/

function getHTTPContentType() {
```

```
            return 'text/xml';
        }
        /**
        * gets the HTTP content type charset for the current response.
        * returns false for non-text content types.
        *
        * Note: getHTTPBody must be called before this.
        *
        * @return string the HTTP content type charset for the current response.
        * @access private
        */
        function getHTTPContentTypeCharset() {
            return $this->soap_defencoding;
        }

        /**
        * add a method to the dispatch map (this has been replaced by the register
method)
        *
        * @param    string $methodname
```

```
        * @param    string $in array of input values
        * @param    string $out array of output values
        * @access   public
        * @deprecated
        */
        function add_to_map($methodname,$in,$out){
                        $this->operations[$methodname] = array('name' =>
$methodname,'in' => $in,'out' => $out);
        }
        /**
        * register a service function with the server
        *
        * @param    string $name the name of the PHP function, class.method or
class..method
        * @param    array $in assoc array of input values: key = param name, value =
param type
        * @param    array $out assoc array of output values: key = param name, value
= param type
        * @param      mixed $namespace the element namespace for the method or
false
        * @param      mixed $soapaction the soapaction for the method or false
```

```
        * @param       mixed $style optional (rpc|document) or false Note: when
'document' is specified, parameter and return wrappers are created for you
automatically

        * @param       mixed $use optional (encoded|literal) or false

        * @param       string $documentation optional Description to include in
WSDL

        * @param       string $encodingStyle optional (usually
'http://schemas.xmlsoap.org/soap/encoding/' for encoded)

        * @access   public

        */

        function
register($name,$in=array(),$out=array(),$namespace=false,$soapaction=false,$style=
false,$use=false,$documentation='',$encodingStyle=''){

                global $HTTP_SERVER_VARS;

                if($this->externalWSDLURL){

                        die('You cannot bind to an external WSDL file, and register
methods outside of it! Please choose either WSDL or no WSDL.');

                }

                if (! $name) {

                        die('You must specify a name when you register an operation');

                }

                if (!is_array($in)) {
```

```
                die('You must provide an array for operation inputs');

        }

        if (!is_array($out)) {

                die('You must provide an array for operation outputs');

        }

        if(false == $namespace) {

        }

        if(false == $soapaction) {

                if (isset($_SERVER)) {

                        $SERVER_NAME = $_SERVER['SERVER_NAME'];

                        $SCRIPT_NAME = isset($_SERVER['PHP_SELF']) ?
$_SERVER['PHP_SELF'] : $_SERVER['SCRIPT_NAME'];

                        $HTTPS = isset($_SERVER['HTTPS']) ?
$_SERVER['HTTPS'] : (isset($HTTP_SERVER_VARS['HTTPS']) ?
$HTTP_SERVER_VARS['HTTPS'] : 'off');

                } elseif (isset($HTTP_SERVER_VARS)) {

                        $SERVER_NAME =
$HTTP_SERVER_VARS['SERVER_NAME'];

                        $SCRIPT_NAME =
isset($HTTP_SERVER_VARS['PHP_SELF']) ?
$HTTP_SERVER_VARS['PHP_SELF'] :
$HTTP_SERVER_VARS['SCRIPT_NAME'];
```

```
                        $HTTPS = isset($HTTP_SERVER_VARS['HTTPS']) ?
$HTTP_SERVER_VARS['HTTPS'] : 'off';
                    } else {
                        $this->setError("Neither _SERVER nor
HTTP_SERVER_VARS is available");
                    }
        if ($HTTPS == '1' || $HTTPS == 'on') {
            $SCHEME = 'https';
        } else {
            $SCHEME = 'http';
        }
                    $soapaction =
"$SCHEME://$SERVER_NAME$SCRIPT_NAME/$name";
                }
                if(false == $style) {
                    $style = "rpc";
                }
                if(false == $use) {
                    $use = "encoded";
                }
                if ($use == 'encoded' && $encodingStyle == '') {
```

```
                $encodingStyle = 'http://schemas.xmlsoap.org/soap/encoding/';
            }

            $this->operations[$name] = array(
        'name' => $name,
        'in' => $in,
        'out' => $out,
        'namespace' => $namespace,
        'soapaction' => $soapaction,
        'style' => $style);
    if($this->wsdl){
      $this->wsdl-
>addOperation($name,$in,$out,$namespace,$soapaction,$style,$use,$documentation,
$encodingStyle);
        }
            return true;
    }

    /**
    * Specify a fault to be returned to the client.
    * This also acts as a flag to the server that a fault has occured.
```

```
        *
        * @param      string $faultcode
        * @param      string $faultstring
        * @param      string $faultactor
        * @param      string $faultdetail
        * @access   public
        */
        function fault($faultcode,$faultstring,$faultactor='',$faultdetail=''){
                if ($faultdetail == '' && $this->debug_flag) {
                        $faultdetail = $this->getDebug();
                }
                $this->fault = new
nusoap_fault($faultcode,$faultactor,$faultstring,$faultdetail);
                $this->fault->soap_defencoding = $this->soap_defencoding;
        }


    /**
    * Sets up wsdl object.
    * Acts as a flag to enable internal WSDL generation
    *
```

```
    * @param string $serviceName, name of the service
    * @param mixed $namespace optional 'tns' service namespace or false
    * @param mixed $endpoint optional URL of service endpoint or false
    * @param string $style optional (rpc|document) WSDL style (also specified by operation)
    * @param string $transport optional SOAP transport
    * @param mixed $schemaTargetNamespace optional 'types' targetNamespace for service schema or false
    */
    function configureWSDL($serviceName,$namespace = false,$endpoint = false,$style='rpc', $transport = 'http://schemas.xmlsoap.org/soap/http', $schemaTargetNamespace = false)
    {
        global $HTTP_SERVER_VARS;

            if (isset($_SERVER)) {
                $SERVER_NAME = $_SERVER['SERVER_NAME'];
                $SERVER_PORT = $_SERVER['SERVER_PORT'];
                $SCRIPT_NAME = isset($_SERVER['PHP_SELF']) ? $_SERVER['PHP_SELF'] : $_SERVER['SCRIPT_NAME'];
```

```
                $HTTPS = isset($_SERVER['HTTPS']) ?
$_SERVER['HTTPS'] : (isset($HTTP_SERVER_VARS['HTTPS']) ?
$HTTP_SERVER_VARS['HTTPS'] : 'off');

            } elseif (isset($HTTP_SERVER_VARS)) {

                $SERVER_NAME =
$HTTP_SERVER_VARS['SERVER_NAME'];

                $SERVER_PORT =
$HTTP_SERVER_VARS['SERVER_PORT'];

                $SCRIPT_NAME =
isset($HTTP_SERVER_VARS['PHP_SELF']) ?
$HTTP_SERVER_VARS['PHP_SELF'] :
$HTTP_SERVER_VARS['SCRIPT_NAME'];

                $HTTPS = isset($HTTP_SERVER_VARS['HTTPS']) ?
$HTTP_SERVER_VARS['HTTPS'] : 'off';

            } else {

                $this->setError("Neither _SERVER nor
HTTP_SERVER_VARS is available");

            }

            // If server name has port number attached then strip it (else port
number gets duplicated in WSDL output) (occurred using lighttpd and FastCGI)

            $colon = strpos($SERVER_NAME,":");

            if ($colon) {

              $SERVER_NAME = substr($SERVER_NAME, 0, $colon);
```

```
                }
                if ($SERVER_PORT == 80) {
                        $SERVER_PORT = '';
                } else {
                        $SERVER_PORT = ':' . $SERVER_PORT;
                }
        if(false == $namespace) {
           $namespace = "http://$SERVER_NAME/soap/$serviceName";
        }

        if(false == $endpoint) {
           if ($HTTPS == '1' || $HTTPS == 'on') {
                        $SCHEME = 'https';
           } else {
                        $SCHEME = 'http';
           }
           $endpoint =
"$SCHEME://$SERVER_NAME$SERVER_PORT$SCRIPT_NAME";
        }
        if(false == $schemaTargetNamespace) {
```

```
        $schemaTargetNamespace = $namespace;
    }
            $this->wsdl = new wsdl;
            $this->wsdl->serviceName = $serviceName;
    $this->wsdl->endpoint = $endpoint;
            $this->wsdl->namespaces['tns'] = $namespace;
            $this->wsdl->namespaces['soap'] =
'http://schemas.xmlsoap.org/wsdl/soap/';
            $this->wsdl->namespaces['wsdl'] = 'http://schemas.xmlsoap.org/wsdl/';
            if ($schemaTargetNamespace != $namespace) {
                $this->wsdl->namespaces['types'] = $schemaTargetNamespace;
            }
    $this->wsdl->schemas[$schemaTargetNamespace][0] = new
nusoap_xmlschema('', '', $this->wsdl->namespaces);
    if ($style == 'document') {
            $this->wsdl->schemas[$schemaTargetNamespace][0]-
>schemaInfo['elementFormDefault'] = 'qualified';
    }
    $this->wsdl->schemas[$schemaTargetNamespace][0]-
>schemaTargetNamespace = $schemaTargetNamespace;
```

```
        $this->wsdl->schemas[$schemaTargetNamespace][0]-
>imports['http://schemas.xmlsoap.org/soap/encoding/'][0] = array('location' => '',
'loaded' => true);

        $this->wsdl->schemas[$schemaTargetNamespace][0]-
>imports['http://schemas.xmlsoap.org/wsdl/'][0] = array('location' => '', 'loaded' =>
true);

        $this->wsdl->bindings[$serviceName.'Binding'] = array(

            'name'=>$serviceName.'Binding',

            'style'=>$style,

            'transport'=>$transport,

            'portType'=>$serviceName.'PortType');

        $this->wsdl->ports[$serviceName.'Port'] = array(

            'binding'=>$serviceName.'Binding',

            'location'=>$endpoint,

            'bindingType'=>'http://schemas.xmlsoap.org/wsdl/soap/');

    }

}

/**

 * Backward compatibility

 */

class soap_server extends nusoap_server {
```

```
}
?>
<?php
/**
* parses a WSDL file, allows access to it's data, other utility methods.
* also builds WSDL structures programmatically.
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access public
*/
class wsdl extends nusoap_base {
        // URL or filename of the root of this WSDL
    var $wsdl;
    // define internal arrays of bindings, ports, operations, messages, etc.
    var $schemas = array();
    var $currentSchema;
    var $message = array();
    var $complexTypes = array();
```

```
var $messages = array();
var $currentMessage;
var $currentOperation;
var $portTypes = array();
var $currentPortType;
var $bindings = array();
var $currentBinding;
var $ports = array();
var $currentPort;
var $opData = array();
var $status = '';
var $documentation = false;
var $endpoint = '';
// array of wsdl docs to import
var $import = array();
// parser vars
var $parser;
var $position = 0;
var $depth = 0;
var $depth_array = array();
```

```
        // for getting wsdl
        var $proxyhost = '';
    var $proxyport = '';
        var $proxyusername = '';
        var $proxypassword = '';
        var $timeout = 0;
        var $response_timeout = 30;
        var $curl_options = array();   // User-specified cURL options
        var $use_curl = false;                  // whether to always try to use cURL
        // for HTTP authentication
        var $username = '';                     // Username for HTTP
authentication
        var $password = '';                     // Password for HTTP
authentication
        var $authtype = '';                     // Type of HTTP authentication
        var $certRequest = array();             // Certificate for HTTP SSL
authentication
    /**
     * constructor
     *
     * @param string $wsdl WSDL document URL
```

```
            * @param string $proxyhost
            * @param string $proxyport
            * @param string $proxyusername
            * @param string $proxypassword
            * @param integer $timeout set the connection timeout
            * @param integer $response_timeout set the response timeout
            * @param array $curl_options user-specified cURL options
            * @param boolean $use_curl try to use cURL
     * @access public
     */
    function wsdl($wsdl =
",$proxyhost=false,$proxyport=false,$proxyusername=false,$proxypassword=false,$t
imeout=0,$response_timeout=30,$curl_options=null,$use_curl=false){
                    parent::nusoap_base();
                    $this->debug("ctor wsdl=$wsdl timeout=$timeout
response_timeout=$response_timeout");
        $this->proxyhost = $proxyhost;
        $this->proxyport = $proxyport;
                    $this->proxyusername = $proxyusername;
                    $this->proxypassword = $proxypassword;
                    $this->timeout = $timeout;
```

```
                $this->response_timeout = $response_timeout;
                if (is_array($curl_options))
                        $this->curl_options = $curl_options;
                $this->use_curl = $use_curl;
                $this->fetchWSDL($wsdl);
}

        /**
         * fetches the WSDL document and parses it
         *
         * @access public
         */
        function fetchWSDL($wsdl) {
                $this->debug("parse and process WSDL path=$wsdl");
                $this->wsdl = $wsdl;
    // parse wsdl file
    if ($this->wsdl != "") {
        $this->parseWSDL($this->wsdl);
    }
    // imports
```

```
        // TODO: handle imports more properly, grabbing them in-line and nesting them
            $imported_urls = array();
            $imported = 1;
            while ($imported > 0) {
                    $imported = 0;
                    // Schema imports
                    foreach ($this->schemas as $ns => $list) {
                            foreach ($list as $xs) {
                                    $wsdlparts = parse_url($this->wsdl);            //
this is bogusly simple!
                                    foreach ($xs->imports as $ns2 => $list2) {
                                       for ($ii = 0; $ii < count($list2); $ii++) {
                                          if (! $list2[$ii]['loaded']) {
                                             $this->schemas[$ns]-
>imports[$ns2][$ii]['loaded'] = true;
                                             $url = $list2[$ii]['location'];
                                                if ($url != '') {
                                                   $urlparts =
parse_url($url);
                                                   if
(!isset($urlparts['host'])) {
```

```
                                                                $url =
$wdslparts['scheme'] . '://' . $wdslparts['host'] . (isset($wdslparts['port']) ? ':'
.$wdslparts['port'] : '') .

        substr($wdslparts['path'],0,strrpos($wdslparts['path'],'/') + 1) .$urlparts['path'];

                                                                }

                                                                if (! in_array($url,
$imported_urls)) {

                                                        $this->parseWSDL($url);

                                                        $imported++;

                                                        $imported_urls[] = $url;

                                                }

                                                                } else {

                                                                $this-
>debug("Unexpected scenario: empty URL for unloaded import");

                                                                }

                                                        }

                                                }

                        }

                        }

                }

                // WSDL imports
```

```
                    $wsdlparts = parse_url($this->wsdl);          // this is bogusly
simple!
        foreach ($this->import as $ns => $list) {
          for ($ii = 0; $ii < count($list); $ii++) {
              if (! $list[$ii]['loaded']) {
                    $this->import[$ns][$ii]['loaded'] = true;
                    $url = $list[$ii]['location'];
                                        if ($url != '') {
                                            $urlparts = parse_url($url);
                                            if (!isset($urlparts['host'])) {
                                                $url =
$wsdlparts['scheme'] . '://' . $wsdlparts['host'] . (isset($wsdlparts['port']) ? ':' .
$wsdlparts['port'] : '') .
        substr($wsdlparts['path'],0,strrpos($wsdlparts['path'],'/') + 1) .$urlparts['path'];
                                            }
                                            if (! in_array($url,
$imported_urls)) {
                                    $this->parseWSDL($url);
                                    $imported++;
                                    $imported_urls[] = $url;
                                }
```

```
                                        } else {
                                            $this->debug("Unexpected
scenario: empty URL for unloaded import");
                                        }
                                    }
                                }
            }
                    }
        // add new data to operation data
        foreach($this->bindings as $binding => $bindingData) {
            if (isset($bindingData['operations']) && is_array($bindingData['operations']))
{
                foreach($bindingData['operations'] as $operation => $data) {
                    $this->debug('post-parse data gathering for ' . $operation);
                    $this->bindings[$binding]['operations'][$operation]['input'] =
                                        isset($this-
>bindings[$binding]['operations'][$operation]['input']) ?
                                        array_merge($this-
>bindings[$binding]['operations'][$operation]['input'], $this->portTypes[
$bindingData['portType'] ][$operation]['input']) :
                                        $this->portTypes[
$bindingData['portType'] ][$operation]['input'];
```

```
            $this->bindings[$binding]['operations'][$operation]['output'] =
                                        isset($this-
>bindings[$binding]['operations'][$operation]['output']) ?
                                        array_merge($this-
>bindings[$binding]['operations'][$operation]['output'], $this->portTypes[
$bindingData['portType'] ][$operation]['output']) :
                                        $this->portTypes[
$bindingData['portType'] ][$operation]['output'];
            if(isset($this->messages[ $this-
>bindings[$binding]['operations'][$operation]['input']['message'] ])){
                                        $this-
>bindings[$binding]['operations'][$operation]['input']['parts'] = $this->messages[
$this->bindings[$binding]['operations'][$operation]['input']['message'] ];
                                    }
                                    if(isset($this->messages[ $this-
>bindings[$binding]['operations'][$operation]['output']['message'] ])){
                $this-
>bindings[$binding]['operations'][$operation]['output']['parts'] = $this->messages[
$this->bindings[$binding]['operations'][$operation]['output']['message'] ];
            }
            // Set operation style if necessary, but do not override one already
provided
                                    if (isset($bindingData['style']) && !isset($this-
>bindings[$binding]['operations'][$operation]['style'])) {
```

```
                $this->bindings[$binding]['operations'][$operation]['style'] =
$bindingData['style'];
            }
            $this->bindings[$binding]['operations'][$operation]['transport'] =
isset($bindingData['transport']) ? $bindingData['transport'] : '';
            $this->bindings[$binding]['operations'][$operation]['documentation'] =
isset($this->portTypes[ $bindingData['portType'] ][$operation]['documentation']) ?
$this->portTypes[ $bindingData['portType'] ][$operation]['documentation'] : '';
            $this->bindings[$binding]['operations'][$operation]['endpoint'] =
isset($bindingData['endpoint']) ? $bindingData['endpoint'] : '';
          }
        }
      }
        }
  /**
   * parses the wsdl document
   *
   * @param string $wsdl path or URL
   * @access private
   */
  function parseWSDL($wsdl = '') {
            $this->debug("parse WSDL at path=$wsdl");
```

```
        if ($wsdl == '') {
            $this->debug('no wsdl passed to parseWSDL()!!');
            $this->setError('no wsdl passed to parseWSDL()!!');
            return false;
        }
        // parse $wsdl for url format
        $wsdl_props = parse_url($wsdl);
        if (isset($wsdl_props['scheme']) && ($wsdl_props['scheme'] == 'http' ||
$wsdl_props['scheme'] == 'https')) {
            $this->debug('getting WSDL http(s) URL ' . $wsdl);
            // get wsdl
                $tr = new soap_transport_http($wsdl, $this->curl_options, $this-
>use_curl);
                        $tr->request_method = 'GET';
                        $tr->useSOAPAction = false;
                        if($this->proxyhost && $this->proxyport){
                                $tr->setProxy($this->proxyhost,$this->proxyport,$this-
>proxyusername,$this->proxypassword);
                        }
                        if ($this->authtype != '') {
```

```
                        $tr->setCredentials($this->username, $this->password,
$this->authtype, array(), $this->certRequest);
                    }
                    $tr->setEncoding('gzip, deflate');
                    $wsdl_string = $tr->send('', $this->timeout, $this-
>response_timeout);
                    //$this->debug("WSDL request\n" . $tr->outgoing_payload);
                    //$this->debug("WSDL response\n" . $tr->incoming_payload);
                    $this->appendDebug($tr->getDebug());
                    // catch errors
                    if($err = $tr->getError() ){
                        $errstr = 'Getting ' . $wsdl . ' - HTTP ERROR: '.$err;
                        $this->debug($errstr);
                $this->setError($errstr);
                        unset($tr);
                return false;
                    }
                    unset($tr);
                    $this->debug("got WSDL URL");
        } else {
            // $wsdl is not http(s), so treat it as a file URL or plain file path
```

```
        if (isset($wsdl_props['scheme']) && ($wsdl_props['scheme'] == 'file') &&
isset($wsdl_props['path'])) {
                $path = isset($wsdl_props['host']) ? ($wsdl_props['host'] . ':' .
$wsdl_props['path']) : $wsdl_props['path'];
        } else {
                $path = $wsdl;
        }
        $this->debug('getting WSDL file ' . $path);
        if ($fp = @fopen($path, 'r')) {
           $wsdl_string = '';
           while ($data = fread($fp, 32768)) {
              $wsdl_string .= $data;
           }
           fclose($fp);
        } else {
                $errstr = "Bad path to WSDL file $path";
                $this->debug($errstr);
           $this->setError($errstr);
           return false;
        }
      }
```

```
$this->debug('Parse WSDL');
// end new code added
// Create an XML parser.
$this->parser = xml_parser_create();
// Set the options for parsing the XML data.
// xml_parser_set_option($parser, XML_OPTION_SKIP_WHITE, 1);
xml_parser_set_option($this->parser, XML_OPTION_CASE_FOLDING, 0);
// Set the object for the parser.
xml_set_object($this->parser, $this);
// Set the element handlers for the parser.
xml_set_element_handler($this->parser, 'start_element', 'end_element');
xml_set_character_data_handler($this->parser, 'character_data');
// Parse the XML file.
if (!xml_parse($this->parser, $wsdl_string, true)) {
   // Display an error message.
   $errstr = sprintf(
                              'XML error parsing WSDL from %s on line %d: %s',
                              $wsdl,
      xml_get_current_line_number($this->parser),
      xml_error_string(xml_get_error_code($this->parser))
```

```
            );
        $this->debug($errstr);
                        $this->debug("XML payload:\n" . $wsdl_string);
        $this->setError($errstr);
        return false;
    }
                // free the parser
    xml_parser_free($this->parser);
    $this->debug('Parsing WSDL done');
                // catch wsdl parse errors
                if($this->getError()){
                        return false;
                }
    return true;
}
/**
 * start-element handler
 *
 * @param string $parser XML parser object
 * @param string $name element name
```

```
     * @param string $attrs associative array of attributes
     * @access private
     */
    function start_element($parser, $name, $attrs)
    {
        if ($this->status == 'schema') {
            $this->currentSchema->schemaStartElement($parser, $name, $attrs);
            $this->appendDebug($this->currentSchema->getDebug());
            $this->currentSchema->clearDebug();
        } elseif (preg_match('/schema$/', $name)) {
            $this->debug('Parsing WSDL schema');
            // $this->debug("startElement for $name ($attrs[name]). status = $this->status
(".$this->getLocalPart($name).")");
            $this->status = 'schema';
            $this->currentSchema = new nusoap_xmlschema('', '', $this->namespaces);
            $this->currentSchema->schemaStartElement($parser, $name, $attrs);
            $this->appendDebug($this->currentSchema->getDebug());
            $this->currentSchema->clearDebug();
        } else {
            // position in the total number of elements, starting from 0
```

```
$pos = $this->position++;
$depth = $this->depth++;
// set self as current value for this depth
$this->depth_array[$depth] = $pos;
$this->message[$pos] = array('cdata' => '');
// process attributes
if (count($attrs) > 0) {
                        // register namespace declarations
   foreach($attrs as $k => $v) {
      if (preg_match('/^xmlns/',$k)) {
         if ($ns_prefix = substr(strrchr($k, ':'), 1)) {
            $this->namespaces[$ns_prefix] = $v;
         } else {
            $this->namespaces['ns' . (count($this->namespaces) + 1)] = $v;
         }
         if ($v == 'http://www.w3.org/2001/XMLSchema' || $v ==
'http://www.w3.org/1999/XMLSchema' || $v ==
'http://www.w3.org/2000/10/XMLSchema') {
            $this->XMLSchemaVersion = $v;
            $this->namespaces['xsi'] = $v . '-instance';
         }
```

```
            }
        }
        // expand each attribute prefix to its namespace
        foreach($attrs as $k => $v) {
            $k = strpos($k, ':') ? $this->expandQname($k) : $k;
            if ($k != 'location' && $k != 'soapAction' && $k != 'namespace') {
                $v = strpos($v, ':') ? $this->expandQname($v) : $v;
            }
            $eAttrs[$k] = $v;
        }
        $attrs = $eAttrs;
    } else {
        $attrs = array();
    }
    // get element prefix, namespace and name
    if (preg_match('/:/', $name)) {
        // get ns prefix
        $prefix = substr($name, 0, strpos($name, ':'));
        // get ns
```

```
            $namespace = isset($this->namespaces[$prefix]) ? $this-
>namespaces[$prefix] : '';
            // get unqualified name
            $name = substr(strstr($name, ':'), 1);
        }
                        // process attributes, expanding any prefixes to namespaces
        // find status, register data
        switch ($this->status) {
            case 'message':
                if ($name == 'part') {
                                    if (isset($attrs['type'])) {
                                $this->debug("msg " . $this->currentMessage . ": found
part (with type) $attrs[name]: " . implode(',', $attrs));
                                $this->messages[$this->currentMessage][$attrs['name']] =
$attrs['type'];
                                    }
                                    if (isset($attrs['element'])) {
                                $this->debug("msg " . $this->currentMessage . ": found
part (with element) $attrs[name]: " . implode(',', $attrs));
                                    $this->messages[$this-
>currentMessage][$attrs['name']] = $attrs['element'] . '^';
                                    }
```

```
                }
                break;
                    case 'portType':
                        switch ($name) {
                            case 'operation':
                                $this->currentPortOperation = $attrs['name'];
                                $this->debug("portType $this->currentPortType
operation: $this->currentPortOperation");
                                if (isset($attrs['parameterOrder'])) {
                                    $this->portTypes[$this-
>currentPortType][$attrs['name']]['parameterOrder'] = $attrs['parameterOrder'];
                                    }
                                    break;
                                case 'documentation':
                                    $this->documentation = true;
                                    break;
                                // merge input/output data
                                default:
                                    $m = isset($attrs['message']) ? $this-
>getLocalPart($attrs['message']) : '';
```

```
                                        $this->portTypes[$this-
>currentPortType][$this->currentPortOperation][$name]['message'] = $m;
                                        break;
                                    }
                                break;
                                case 'binding':
                                    switch ($name) {
                                        case 'binding':
                                            // get ns prefix
                                            if (isset($attrs['style'])) {
                                            $this->bindings[$this->currentBinding]['prefix']
= $prefix;
                                            }
                                            $this->bindings[$this->currentBinding] =
array_merge($this->bindings[$this->currentBinding], $attrs);
                                            break;
                                            case 'header':
                                                $this->bindings[$this-
>currentBinding]['operations'][$this->currentOperation][$this->opStatus]['headers'][]
= $attrs;
                                                break;
                                            case 'operation':
```

```
                                        if (isset($attrs['soapAction'])) {
                                            $this->bindings[$this->currentBinding]['operations'][$this->currentOperation]['soapAction'] = $attrs['soapAction'];
                                        }
                                        if (isset($attrs['style'])) {
                                            $this->bindings[$this->currentBinding]['operations'][$this->currentOperation]['style'] = $attrs['style'];
                                        }
                                        if (isset($attrs['name'])) {
                                            $this->currentOperation = $attrs['name'];
                                            $this->debug("current binding operation: $this->currentOperation");
                                            $this->bindings[$this->currentBinding]['operations'][$this->currentOperation]['name'] = $attrs['name'];
                                            $this->bindings[$this->currentBinding]['operations'][$this->currentOperation]['binding'] = $this->currentBinding;
                                            $this->bindings[$this->currentBinding]['operations'][$this->currentOperation]['endpoint'] = isset($this->bindings[$this->currentBinding]['endpoint']) ? $this->bindings[$this->currentBinding]['endpoint'] : '';
                                        }
```

```
                    break;
                case 'input':
                    $this->opStatus = 'input';
                    break;
                case 'output':
                    $this->opStatus = 'output';
                    break;
                case 'body':
                    if (isset($this->bindings[$this->currentBinding]['operations'][$this->currentOperation][$this->opStatus])) {
                        $this->bindings[$this->currentBinding]['operations'][$this->currentOperation][$this->opStatus] = array_merge($this->bindings[$this->currentBinding]['operations'][$this->currentOperation][$this->opStatus], $attrs);
                    } else {
                        $this->bindings[$this->currentBinding]['operations'][$this->currentOperation][$this->opStatus] = $attrs;
                    }
                    break;
            }
            break;
        case 'service':
```

```
                                        switch ($name) {
                                            case 'port':
                                                $this->currentPort = $attrs['name'];
                                                $this->debug('current port: ' . $this-
>currentPort);
                                                $this->ports[$this->currentPort]['binding']
= $this->getLocalPart($attrs['binding']);

                                                break;
                                            case 'address':
                                                $this->ports[$this->currentPort]['location']
= $attrs['location'];
                                                $this->ports[$this-
>currentPort]['bindingType'] = $namespace;
                                                $this->bindings[ $this->ports[$this-
>currentPort]['binding'] ]['bindingType'] = $namespace;
                                                $this->bindings[ $this->ports[$this-
>currentPort]['binding'] ]['endpoint'] = $attrs['location'];
                                                break;
                                        }
                                        break;
                                }
```

```
                // set status
                switch ($name) {
                        case 'import':
                            if (isset($attrs['location'])) {
                $this->import[$attrs['namespace']][] = array('location' =>
$attrs['location'], 'loaded' => false);
                $this->debug('parsing import ' . $attrs['namespace']. ' - ' . $attrs['location']
. ' (' . count($this->import[$attrs['namespace']]).')');
                                } else {
                $this->import[$attrs['namespace']][] = array('location' => '', 'loaded' =>
true);
                                        if (! $this-
>getPrefixFromNamespace($attrs['namespace'])) {
                                                $this->namespaces['ns'.(count($this-
>namespaces)+1)] = $attrs['namespace'];
                                        }
                $this->debug('parsing import ' . $attrs['namespace']. ' - [no location] (' .
count($this->import[$attrs['namespace']]).')');
                                }
                                break;
                        //wait for schema
                        //case 'types':
```

```
                    //          $this->status = 'schema';
                    //          break;
                    case 'message':
                              $this->status = 'message';
                              $this->messages[$attrs['name']] = array();
                              $this->currentMessage = $attrs['name'];
                              break;
                    case 'portType':
                              $this->status = 'portType';
                              $this->portTypes[$attrs['name']] = array();
                              $this->currentPortType = $attrs['name'];
                              break;
                    case "binding":
                              if (isset($attrs['name'])) {
                              // get binding name
                                        if (strpos($attrs['name'], ':')) {
                                        $this->currentBinding = $this-
>getLocalPart($attrs['name']);
                                        } else {
                                        $this->currentBinding = $attrs['name'];
```

```
                                        }

                                        $this->status = 'binding';

                                        $this->bindings[$this-
>currentBinding]['portType'] = $this->getLocalPart($attrs['type']);

                                        $this->debug("current binding: $this-
>currentBinding of portType: " . $attrs['type']);

                                }

                                break;

                        case 'service':

                                $this->serviceName = $attrs['name'];

                                $this->status = 'service';

                                $this->debug('current service: ' . $this->serviceName);

                                break;

                        case 'definitions':

                                foreach ($attrs as $name => $value) {

                                        $this->wsdl_info[$name] = $value;

                                }

                                break;

                        }

                }

        }
```

```
        /**
        * end-element handler
        *
        * @param string $parser XML parser object
        * @param string $name element name
        * @access private
        */
        function end_element($parser, $name){
                // unset schema status
                if (/*preg_match('/types$/', $name) ||*/ preg_match('/schema$/',
$name)) {
                        $this->status = "";
        $this->appendDebug($this->currentSchema->getDebug());
        $this->currentSchema->clearDebug();
                        $this->schemas[$this->currentSchema-
>schemaTargetNamespace][] = $this->currentSchema;
        $this->debug('Parsing WSDL schema done');
                }
                if ($this->status == 'schema') {
                        $this->currentSchema->schemaEndElement($parser, $name);
                } else {
```

```
                    // bring depth down a notch
                    $this->depth--;
                }
                // end documentation
                if ($this->documentation) {
                    //TODO: track the node to which documentation should be assigned; it can be a part, message, etc.
                    //$this->portTypes[$this->currentPortType][$this->currentPortOperation]['documentation'] = $this->documentation;
                    $this->documentation = false;
                }
            }
            /**
             * element content handler
             *
             * @param string $parser XML parser object
             * @param string $data element content
             * @access private
             */
            function character_data($parser, $data)
            {
```

```
                $pos = isset($this->depth_array[$this->depth]) ? $this->depth_array[$this->depth] : 0;
                if (isset($this->message[$pos]['cdata'])) {
                        $this->message[$pos]['cdata'] .= $data;
                }
                if ($this->documentation) {
                        $this->documentation .= $data;
                }
        }
        /**
        * if authenticating, set user credentials here
        *
        * @param    string $username
        * @param    string $password
        * @param       string $authtype (basic|digest|certificate|ntlm)
        * @param       array $certRequest (keys must be cainfofile (optional), sslcertfile, sslkeyfile, passphrase, certpassword (optional), verifypeer (optional), verifyhost (optional): see corresponding options in cURL docs)
        * @access   public
        */
```

```
        function setCredentials($username, $password, $authtype = 'basic',
$certRequest = array()) {
                $this->debug("setCredentials username=$username
authtype=$authtype certRequest=");
                $this->appendDebug($this->varDump($certRequest));
                $this->username = $username;
                $this->password = $password;
                $this->authtype = $authtype;
                $this->certRequest = $certRequest;
        }

        function getBindingData($binding)
        {
                if (is_array($this->bindings[$binding])) {
                        return $this->bindings[$binding];
                }
        }
        /**
         * returns an assoc array of operation names => operation data
         *
         * @param string $portName WSDL port name
```

```
     * @param string $bindingType eg: soap, smtp, dime (only soap and soap12 are currently supported)
     * @return array
     * @access public
     */
    function getOperations($portName = '', $bindingType = 'soap') {
        $ops = array();
        if ($bindingType == 'soap') {
            $bindingType = 'http://schemas.xmlsoap.org/wsdl/soap/';
        } elseif ($bindingType == 'soap12') {
            $bindingType = 'http://schemas.xmlsoap.org/wsdl/soap12/';
        } else {
            $this->debug("getOperations bindingType $bindingType may not be supported");
        }
        $this->debug("getOperations for port '$portName' bindingType $bindingType");
        // loop thru ports
        foreach($this->ports as $port => $portData) {
            $this->debug("getOperations checking port $port bindingType " . $portData['bindingType']);
```

```
                if ($portName == '' || $port == $portName) {
                    // binding type of port matches parameter
                    if ($portData['bindingType'] == $bindingType) {
                        $this->debug("getOperations found port $port bindingType $bindingType");
                        //$this->debug("port data: " . $this->varDump($portData));
                        //$this->debug("bindings: " . $this->varDump($this->bindings[ $portData['binding'] ]));
                        // merge bindings
                        if (isset($this->bindings[ $portData['binding'] ]['operations'])) {
                            $ops = array_merge ($ops, $this->bindings[ $portData['binding'] ]['operations']);
                        }
                    }
                }
            }
            if (count($ops) == 0) {
                $this->debug("getOperations found no operations for port '$portName' bindingType $bindingType");
            }
```

```
        return $ops;
}
/**
 * returns an associative array of data necessary for calling an operation
 *
 * @param string $operation name of operation
 * @param string $bindingType type of binding eg: soap, soap12
 * @return array
 * @access public
 */
function getOperationData($operation, $bindingType = 'soap')
{
        if ($bindingType == 'soap') {
                $bindingType = 'http://schemas.xmlsoap.org/wsdl/soap/';
        } elseif ($bindingType == 'soap12') {
                $bindingType = 'http://schemas.xmlsoap.org/wsdl/soap12/';
        }
        // loop thru ports
        foreach($this->ports as $port => $portData) {
                // binding type of port matches parameter
```

```
                    if ($portData['bindingType'] == $bindingType) {
                        // get binding
                        //foreach($this->bindings[ $portData['binding'] ]['operations'] as $bOperation => $opData) {
                        foreach(array_keys($this->bindings[ $portData['binding'] ]['operations']) as $bOperation) {
                            // note that we could/should also check the namespace here
                            if ($operation == $bOperation) {
                                $opData = $this->bindings[ $portData['binding'] ]['operations'][$operation];
                                return $opData;
                            }
                        }
                    }
                }
            }
            /**
             * returns an associative array of data necessary for calling an operation
             *
             * @param string $soapAction soapAction for operation
```

```
     * @param string $bindingType type of binding eg: soap, soap12
     * @return array
     * @access public
     */
    function getOperationDataForSoapAction($soapAction, $bindingType =
'soap') {
        if ($bindingType == 'soap') {
            $bindingType = 'http://schemas.xmlsoap.org/wsdl/soap/';
        } elseif ($bindingType == 'soap12') {
            $bindingType = 'http://schemas.xmlsoap.org/wsdl/soap12/';
        }
        // loop thru ports
        foreach($this->ports as $port => $portData) {
            // binding type of port matches parameter
            if ($portData['bindingType'] == $bindingType) {
                // loop through operations for the binding
                foreach ($this->bindings[ $portData['binding']
]['operations'] as $bOperation => $opData) {
                    if ($opData['soapAction'] == $soapAction) {
                        return $opData;
                    }
```

```
                                }
                            }
                        }
                }
        /**
* returns an array of information about a given type
* returns false if no type exists by the given name
*
        *          typeDef = array(
        *          'elements' => array(), // refs to elements array
        *          'restrictionBase' => '',
        *          'phpType' => '',
        *          'order' => '(sequence|all)',
        *          'attrs' => array() // refs to attributes array
        *          )
*
* @param string $type the type
* @param string $ns namespace (not prefix) of the type
* @return mixed
* @access public
```

```
    * @see nusoap_xmlschema
    */
        function getTypeDef($type, $ns) {
                $this->debug("in getTypeDef: type=$type, ns=$ns");
                if ((! $ns) && isset($this->namespaces['tns'])) {
                        $ns = $this->namespaces['tns'];
                        $this->debug("in getTypeDef: type namespace forced to $ns");
                }
                if (!isset($this->schemas[$ns])) {
                        foreach ($this->schemas as $ns0 => $schema0) {
                                if (strcasecmp($ns, $ns0) == 0) {
                                        $this->debug("in getTypeDef: replacing schema
namespace $ns with $ns0");
                                        $ns = $ns0;
                                        break;
                                }
                        }
                }
                if (isset($this->schemas[$ns])) {
                        $this->debug("in getTypeDef: have schema for namespace
$ns");
```

```
                for ($i = 0; $i < count($this->schemas[$ns]); $i++) {
                    $xs = &$this->schemas[$ns][$i];
                    $t = $xs->getTypeDef($type);
                    $this->appendDebug($xs->getDebug());
                    $xs->clearDebug();
                    if ($t) {
                        $this->debug("in getTypeDef: found type
$type");
                        if (!isset($t['phpType'])) {
                            // get info for type to tack onto the
element
                            $uqType = substr($t['type'],
strrpos($t['type'], ':') + 1);
                            $ns = substr($t['type'], 0,
strrpos($t['type'], ':'));
                            $etype = $this->getTypeDef($uqType,
$ns);
                            if ($etype) {
                                $this->debug("found type for
[element] $type:");
                                $this->debug($this-
>varDump($etype));
```

```
                                        if (isset($etype['phpType'])) {
                                            $t['phpType'] =
$etype['phpType'];
                                        }
                                        if (isset($etype['elements'])) {
                                            $t['elements'] =
$etype['elements'];
                                        }
                                        if (isset($etype['attrs'])) {
                                            $t['attrs'] = $etype['attrs'];
                                        }
                                    } else {
                                        $this->debug("did not find type
for [element] $type");
                                    }
                                }
                                return $t;
                            }
                        }
                        $this->debug("in getTypeDef: did not find type $type");
                    } else {
```

```
                    $this->debug("in getTypeDef: do not have schema for
namespace $ns");
                }
                return false;
        }

    /**
    * prints html description of services
    *
    * @access private
    */
    function webDescription(){
        global $HTTP_SERVER_VARS;

                if (isset($_SERVER)) {
                    $PHP_SELF = $_SERVER['PHP_SELF'];
                } elseif (isset($HTTP_SERVER_VARS)) {
                    $PHP_SELF = $HTTP_SERVER_VARS['PHP_SELF'];
                } else {
                    $this->setError("Neither _SERVER nor
HTTP_SERVER_VARS is available");
```

```
                }
                $b = '
                <html><head><title>NuSOAP: '.$this->serviceName.'</title>
                <style type="text/css">
                    body    { font-family: arial; color: #000000; background-color: #ffffff; margin: 0px 0px 0px 0px; }
                    p       { font-family: arial; color: #000000; margin-top: 0px; margin-bottom: 12px; }
                    pre { background-color: silver; padding: 5px; font-family: Courier New; font-size: x-small; color: #000000;}
                    ul      { margin-top: 10px; margin-left: 20px; }
                    li      { list-style-type: none; margin-top: 10px; color: #000000; }
                    .content{
                        margin-left: 0px; padding-bottom: 2em; }
                    .nav {
                        padding-top: 10px; padding-bottom: 10px; padding-left: 15px; font-size: .70em;
                        margin-top: 10px; margin-left: 0px; color: #000000;
                        background-color: #ccccff; width: 20%; margin-left: 20px; margin-top: 20px; }
                    .title {
                        font-family: arial; font-size: 26px; color: #ffffff;
```

```
            background-color: #999999; width: 100%;
            margin-left: 0px; margin-right: 0px;
            padding-top: 10px; padding-bottom: 10px;}
        .hidden {
            position: absolute; visibility: hidden; z-index: 200; left: 250px;
top: 100px;
            font-family: arial; overflow: hidden; width: 600;
            padding: 20px; font-size: 10px; background-color: #999999;
            layer-background-color:#FFFFFF; }
        a,a:active  { color: charcoal; font-weight: bold; }
        a:visited   { color: #666666; font-weight: bold; }
        a:hover     { color: cc3300; font-weight: bold; }
    </style>
    <script language="JavaScript" type="text/javascript">
    <!--
    // POP-UP CAPTIONS...
    function lib_bwcheck(){ //Browsercheck (needed)
        this.ver=navigator.appVersion
        this.agent=navigator.userAgent
        this.dom=document.getElementById?1:0
```

```
        this.opera5=this.agent.indexOf("Opera 5")>-1
        this.ie5=(this.ver.indexOf("MSIE 5")>-1 && this.dom &&
!this.opera5)?1:0;
        this.ie6=(this.ver.indexOf("MSIE 6")>-1 && this.dom &&
!this.opera5)?1:0;
        this.ie4=(document.all && !this.dom && !this.opera5)?1:0;
        this.ie=this.ie4||this.ie5||this.ie6
        this.mac=this.agent.indexOf("Mac")>-1
        this.ns6=(this.dom && parseInt(this.ver) >= 5) ?1:0;
        this.ns4=(document.layers && !this.dom)?1:0;
        this.bw=(this.ie6 || this.ie5 || this.ie4 || this.ns4 || this.ns6 ||
this.opera5)
        return this
    }
    var bw = new lib_bwcheck()
    //Makes crossbrowser object.
    function makeObj(obj){
        this.evnt=bw.dom?
document.getElementById(obj):bw.ie4?document.all[obj]:bw.ns4?document.layers[o
bj]:0;
        if(!this.evnt) return false
        this.css=bw.dom||bw.ie4?this.evnt.style:bw.ns4?this.evnt:0;
```

```
    this.wref=bw.dom||bw.ie4?this.evnt:bw.ns4?this.css.document:0;
    this.writeIt=b_writeIt;
    return this
}
// A unit of measure that will be added when setting the position of a layer.
//var px = bw.ns4||window.opera?"":"px";
function b_writeIt(text){
    if (bw.ns4){this.wref.write(text);this.wref.close()}
    else this.wref.innerHTML = text
}
//Shows the messages
var oDesc;
function popup(divid){
    if(oDesc = new makeObj(divid)){
        oDesc.css.visibility = "visible"
    }
}
function popout(){ // Hides message
    if(oDesc) oDesc.css.visibility = "hidden"
}
```

```
//-->

</script>

</head>

<body>

<div class=content>

	<br><br>

	<div class=title>'.$this->serviceName.'</div>

	<div class=nav>

		<p>View the <a
href="'.$PHP_SELF.'?wsdl">WSDL</a> for the service.

		Click on an operation name to view it&amp;apos;s
details.</p>

		<ul>';

		foreach($this->getOperations() as $op => $data){

		  $b .= "<li><a href='#'
onclick=\"popout();popup('$op')\">$op</a></li>";

		  // create hidden div

		  $b .= "<div id='$op' class='hidden'>

		  <a href='#' onclick='popout()'><font
color='#ffffff'>Close</font></a><br><br>";

		  foreach($data as $donnie => $marie){ // loop through
opdata
```

```
                                        if($donnie == 'input' || $donnie ==
'output'){ // show input/output data
                                            $b .= "<font
color='white'>".ucfirst($donnie).':</font><br>';
                                            foreach($marie as $captain =>
$tenille){ // loop through data
                                                if($captain == 'parts'){ //
loop thru parts
                                                    $b .=
"  $captain:<br>";
                                                //if(is_array($tenille)){
                                                    foreach($tenille as
$joanie => $chachi){
                                                        $b
.= "    $joanie: $chachi<br>";
                                                    }
                                                //}
                                                } else {
                                                    $b .=
"  $captain: $tenille<br>";
                                                }
                                            }
                                        } else {
```

```
                                                $b .= "<font
color='white'>".ucfirst($donnie).":</font> $marie<br>";
                                                }
                                            }
                                                $b .= '</div>';
                                        }
                                        $b .= '
                                        <ul>
                                    </div>
                                </div></body></html>';
                                return $b;
    }
        /**
        * serialize the parsed wsdl
        *
        * @param mixed $debug whether to put debug=1 in endpoint URL
        * @return string serialization of WSDL
        * @access public
        */
        function serialize($debug = 0)
```

```
{
        $xml = '<?xml version="1.0" encoding="ISO-8859-1"?>';
        $xml .= "\n<definitions";
        foreach($this->namespaces as $k => $v) {
                $xml .= " xmlns:$k=\"$v\"";
        }
        // 10.9.02 - add poulter fix for wsdl and tns declarations
        if (isset($this->namespaces['wsdl'])) {
                $xml .= " xmlns=\"" . $this->namespaces['wsdl'] . "\"";
        }
        if (isset($this->namespaces['tns'])) {
                $xml .= " targetNamespace=\"" . $this->namespaces['tns'] .
"\"";
        }
        $xml .= '>';
        // imports
        if (sizeof($this->import) > 0) {
                foreach($this->import as $ns => $list) {
                        foreach ($list as $ii) {
                                if ($ii['location'] != '') {
```

```
                                        $xml .= '<import location="' .
$ii['location'] . '" namespace="' . $ns . '" />';
                                    } else {
                                        $xml .= '<import namespace="' . $ns . '"
/>';
                                    }
                                }
                            }
                            // types
                            if (count($this->schemas)>=1) {
                                $xml .= "\n<types>\n";
                                foreach ($this->schemas as $ns => $list) {
                                    foreach ($list as $xs) {
                                        $xml .= $xs->serializeSchema();
                                    }
                                }
                                $xml .= '</types>';
                            }
                            // messages
                            if (count($this->messages) >= 1) {
```

```
                    foreach($this->messages as $msgName => $msgParts) {
                        $xml .= "\n<message name=\"" . $msgName . "'>';
                        if(is_array($msgParts)){
                            foreach($msgParts as $partName => $partType)
{
                                // print 'serializing '.$partType.', sv:
'.$this->XMLSchemaVersion.'<br>';
                                if (strpos($partType, ':')) {
                                    $typePrefix = $this-
>getPrefixFromNamespace($this->getPrefix($partType));
                                } elseif (isset($this->typemap[$this-
>namespaces['xsd']][$partType])) {
                                    // print 'checking typemap: '.$this-
>XMLSchemaVersion.'<br>';
                                    $typePrefix = 'xsd';
                                } else {
                                    foreach($this->typemap as $ns =>
$types) {
                                        if (isset($types[$partType])) {
                                            $typePrefix = $this-
>getPrefixFromNamespace($ns);
                                        }
```

```
                }
                if (!isset($typePrefix)) {
                    die("$partType has no
namespace!");
                }
            }
            $ns = $this-
>getNamespaceFromPrefix($typePrefix);
            $localPart = $this-
>getLocalPart($partType);
            $typeDef = $this-
>getTypeDef($localPart, $ns);
            if ($typeDef['typeClass'] == 'element') {
                $elementortype = 'element';
                if (substr($localPart, -1) == '^') {
                    $localPart =
substr($localPart, 0, -1);
                }
            } else {
                $elementortype = 'type';
            }
```

```
                                        $xml .= "\n" . '  <part name="' .
$partName . '" ' . $elementortype . '="' . $typePrefix . ':' . $localPart . '" />';
                                    }
                                }
                                $xml .= '</message>';
                            }
                        }
                        // bindings & porttypes
                        if (count($this->bindings) >= 1) {
                            $binding_xml = '';
                            $portType_xml = '';
                            foreach($this->bindings as $bindingName => $attrs) {
                                $binding_xml .= "\n<binding name=\"" . $bindingName
. '" type="tns:' . $attrs['portType'] . '">';
                                $binding_xml .= "\n" . '  <soap:binding style="' .
$attrs['style'] . '" transport="' . $attrs['transport'] . '"/>';
                                $portType_xml .= "\n<portType name=\"" .
$attrs['portType'] . '">';
                                foreach($attrs['operations'] as $opName => $opParts) {
                                    $binding_xml .= "\n" . '  <operation name="' .
$opName . '">';
```

```
                $binding_xml .= "\n" . '    <soap:operation
soapAction="' . $opParts['soapAction'] . '" style="'. $opParts['style'] . '"/>';

                if (isset($opParts['input']['encodingStyle']) &&
$opParts['input']['encodingStyle'] != '') {

                        $enc_style = ' encodingStyle="' .
$opParts['input']['encodingStyle'] . '"';

                } else {

                        $enc_style = '';

                }

                $binding_xml .= "\n" . '    <input><soap:body
use="' . $opParts['input']['use'] . '" namespace="' . $opParts['input']['namespace'] . '" .
$enc_style . '/></input>';

                if (isset($opParts['output']['encodingStyle']) &&
$opParts['output']['encodingStyle'] != '') {

                        $enc_style = ' encodingStyle="' .
$opParts['output']['encodingStyle'] . '"';

                } else {

                        $enc_style = '';

                }

                $binding_xml .= "\n" . '    <output><soap:body
use="' . $opParts['output']['use'] . '" namespace="' . $opParts['output']['namespace'] .
'" . $enc_style . '/></output>';

                $binding_xml .= "\n" . '  </operation>';
```

```
                                        $portType_xml .= "\n" . '  <operation name="' .
$opParts['name'] . '"';
                                        if (isset($opParts['parameterOrder'])) {
                                            $portType_xml .= ' parameterOrder="' .
$opParts['parameterOrder'] . '"';
                                        }
                                        $portType_xml .= '>';
                                        if(isset($opParts['documentation']) &&
$opParts['documentation'] != '') {
                                                $portType_xml .= "\n" . '
<documentation>' . htmlspecialchars($opParts['documentation']) .
'</documentation>';
                                        }
                                        $portType_xml .= "\n" . '    <input
message="tns:' . $opParts['input']['message'] . '"/>';
                                        $portType_xml .= "\n" . '    <output
message="tns:' . $opParts['output']['message'] . '"/>';
                                        $portType_xml .= "\n" . '  </operation>';
                                    }
                                    $portType_xml .= "\n" . '</portType>';
                                    $binding_xml .= "\n" . '</binding>';
                                }
```

```
                $xml .= $portType_xml . $binding_xml;
            }
            // services
            $xml .= "\n<service name=\"" . $this->serviceName . "'>';
            if (count($this->ports) >= 1) {
                foreach($this->ports as $pName => $attrs) {
                    $xml .= "\n" . '  <port name="' . $pName . '" binding="tns:' . $attrs['binding'] . '">';
                    $xml .= "\n" . '    <soap:address location="' . $attrs['location'] . ($debug ? '?debug=1' : '') . '"/>';
                    $xml .= "\n" . '  </port>';
                }
            }
            $xml .= "\n" . '</service>';
            return $xml . "\n</definitions>";
        }

        /**
         * determine whether a set of parameters are unwrapped
         * when they are expect to be wrapped, Microsoft-style.
         *
```

```
        * @param string $type the type (element name) of the wrapper
        * @param array $parameters the parameter values for the SOAP call
        * @return boolean whether they parameters are unwrapped (and should be wrapped)
        * @access private
        */
        function parametersMatchWrapped($type, &$parameters) {
                $this->debug("in parametersMatchWrapped type=$type, parameters=");
                $this->appendDebug($this->varDump($parameters));
                // split type into namespace:unqualified-type
                if (strpos($type, ':')) {
                        $uqType = substr($type, strrpos($type, ':') + 1);
                        $ns = substr($type, 0, strrpos($type, ':'));
                        $this->debug("in parametersMatchWrapped: got a prefixed type: $uqType, $ns");
                        if ($this->getNamespaceFromPrefix($ns)) {
                                $ns = $this->getNamespaceFromPrefix($ns);
                                $this->debug("in parametersMatchWrapped: expanded prefixed type: $uqType, $ns");
                        }
```

```
                } else {
                        // TODO: should the type be compared to types in XSD, and
the namespace
                        // set to XSD if the type matches?
                        $this->debug("in parametersMatchWrapped: No namespace for
type $type");
                        $ns = '';
                        $uqType = $type;
                }
                // get the type information
                if (!$typeDef = $this->getTypeDef($uqType, $ns)) {
                        $this->debug("in parametersMatchWrapped: $type ($uqType)
is not a supported type.");
                        return false;
                }
                $this->debug("in parametersMatchWrapped: found typeDef=");
                $this->appendDebug($this->varDump($typeDef));
                if (substr($uqType, -1) == '^') {
                        $uqType = substr($uqType, 0, -1);
                }
                $phpType = $typeDef['phpType'];
```

```
                $arrayType = (isset($typeDef['arrayType']) ? $typeDef['arrayType'] :
'');

                $this->debug("in parametersMatchWrapped: uqType: $uqType, ns:
$ns, phptype: $phpType, arrayType: $arrayType");


                // we expect a complexType or element of complexType

                if ($phpType != 'struct') {

                        $this->debug("in parametersMatchWrapped: not a struct");

                        return false;

                }

                // see whether the parameter names match the elements

                if (isset($typeDef['elements']) && is_array($typeDef['elements'])) {

                        $elements = 0;

                        $matches = 0;

                        foreach ($typeDef['elements'] as $name => $attrs) {

                                if (isset($parameters[$name])) {

                                        $this->debug("in parametersMatchWrapped:
have parameter named $name");

                                        $matches++;

                                } else {
```

```
                                        $this->debug("in parametersMatchWrapped: do
not have parameter named $name");
                                }
                                $elements++;
                        }
                        $this->debug("in parametersMatchWrapped: $matches
parameter names match $elements wrapped parameter names");
                        if ($matches == 0) {
                                return false;
                        }
                        return true;
                }
                // since there are no elements for the type, if the user passed no
                // parameters, the parameters match wrapped.
                $this->debug("in parametersMatchWrapped: no elements type
$ns:$uqType");
                return count($parameters) == 0;
        }
        /**
         * serialize PHP values according to a WSDL message definition
         * contrary to the method name, this is not limited to RPC
```

```
        *
        * TODO
        * - multi-ref serialization
        * - validate PHP values against type definitions, return errors if invalid
        *
        * @param string $operation operation name
        * @param string $direction (input|output)
        * @param mixed $parameters parameter value(s)
        * @param string $bindingType (soap|soap12)
        * @return mixed parameters serialized as XML or false on error (e.g. operation not found)
        * @access public
        */
       function serializeRPCParameters($operation, $direction, $parameters, $bindingType = 'soap') {
            $this->debug("in serializeRPCParameters: operation=$operation, direction=$direction, XMLSchemaVersion=$this->XMLSchemaVersion, bindingType=$bindingType");
            $this->appendDebug('parameters=' . $this->varDump($parameters));
            if ($direction != 'input' && $direction != 'output') {
                $this->debug('The value of the \$direction argument needs to be either "input" or "output"');
```

```
                $this->setError('The value of the \$direction argument needs to be either "input" or "output"');
                return false;
            }
            if (!$opData = $this->getOperationData($operation, $bindingType)) {
                $this->debug('Unable to retrieve WSDL data for operation: ' . $operation . ' bindingType: ' . $bindingType);
                $this->setError('Unable to retrieve WSDL data for operation: ' . $operation . ' bindingType: ' . $bindingType);
                return false;
            }
            $this->debug('in serializeRPCParameters: opData:');
            $this->appendDebug($this->varDump($opData));
            // Get encoding style for output and set to current
            $encodingStyle = 'http://schemas.xmlsoap.org/soap/encoding/';
            if(($direction == 'input') && isset($opData['output']['encodingStyle']) && ($opData['output']['encodingStyle'] != $encodingStyle)) {
                $encodingStyle = $opData['output']['encodingStyle'];
                $enc_style = $encodingStyle;
            }
```

```
                // set input params
                $xml = '';
                if (isset($opData[$direction]['parts']) &&
sizeof($opData[$direction]['parts']) > 0) {
                        $parts = &$opData[$direction]['parts'];
                        $part_count = sizeof($parts);
                        $style = $opData['style'];
                        $use = $opData[$direction]['use'];
                        $this->debug("have $part_count part(s) to serialize using
$style/$use");
                        if (is_array($parameters)) {
                                $parametersArrayType = $this-
>isArraySimpleOrStruct($parameters);
                                $parameter_count = count($parameters);
                                $this->debug("have $parameter_count parameter(s)
provided as $parametersArrayType to serialize");
                                // check for Microsoft-style wrapped parameters
                                if ($style == 'document' && $use == 'literal' &&
$part_count == 1 && isset($parts['parameters'])) {
                                        $this->debug('check whether the caller has
wrapped the parameters');
```

```
                    if ($direction == 'output' &&
$parametersArrayType == 'arraySimple' && $parameter_count == 1) {
                        // TODO: consider checking here for
double-wrapping, when
                        // service function wraps, then NuSOAP
wraps again
                        $this->debug("change simple array to
associative with 'parameters' element");
                        $parameters['parameters'] =
$parameters[0];
                        unset($parameters[0]);
                    }
                    if (($parametersArrayType == 'arrayStruct' ||
$parameter_count == 0) && !isset($parameters['parameters'])) {
                        $this->debug('check whether caller\'s
parameters match the wrapped ones');
                        if ($this-
>parametersMatchWrapped($parts['parameters'], $parameters)) {
                            $this->debug('wrap the
parameters for the caller');
                            $parameters = array('parameters'
=> $parameters);
                            $parameter_count = 1;
```

```
                                        }
                                    }
                                }
                                foreach ($parts as $name => $type) {
                                    $this->debug("serializing part $name of type $type");
                                    // Track encoding style
                                    if (isset($opData[$direction]['encodingStyle']) && $encodingStyle != $opData[$direction]['encodingStyle']) {
                                        $encodingStyle = $opData[$direction]['encodingStyle'];
                                        $enc_style = $encodingStyle;
                                    } else {
                                        $enc_style = false;
                                    }
                                    // NOTE: add error handling here
                                    // if serializeType returns false, then catch global error and fault
                                    if ($parametersArrayType == 'arraySimple') {
                                        $p = array_shift($parameters);
                                        $this->debug('calling serializeType w/indexed param');
```

```
                                        $xml .= $this->serializeType($name,
$type, $p, $use, $enc_style);
                                    } elseif (isset($parameters[$name])) {
                                        $this->debug('calling serializeType
w/named param');
                                        $xml .= $this->serializeType($name,
$type, $parameters[$name], $use, $enc_style);
                                    } else {
                                        // TODO: only send nillable
                                        $this->debug('calling serializeType
w/null param');
                                        $xml .= $this->serializeType($name,
$type, null, $use, $enc_style);
                                    }
                                }
                            } else {
                                $this->debug('no parameters passed.');
                            }
                        }
                        $this->debug("serializeRPCParameters returning: $xml");
                        return $xml;
                    }
```

```
        /**
         * serialize a PHP value according to a WSDL message definition
         *
         * TODO
         * - multi-ref serialization
         * - validate PHP values against type definitions, return errors if invalid
         *
         * @param string $operation operation name
         * @param string $direction (input|output)
         * @param mixed $parameters parameter value(s)
         * @return mixed parameters serialized as XML or false on error (e.g.
operation not found)
         * @access public
         * @deprecated
         */
        function serializeParameters($operation, $direction, $parameters)
        {
                $this->debug("in serializeParameters: operation=$operation,
direction=$direction, XMLSchemaVersion=$this->XMLSchemaVersion");
                $this->appendDebug('parameters=' . $this->varDump($parameters));
```

```
                if ($direction != 'input' && $direction != 'output') {
                        $this->debug('The value of the \$direction argument needs to be either "input" or "output"');
                        $this->setError('The value of the \$direction argument needs to be either "input" or "output"');
                        return false;
                }
                if (!$opData = $this->getOperationData($operation)) {
                        $this->debug('Unable to retrieve WSDL data for operation: ' . $operation);
                        $this->setError('Unable to retrieve WSDL data for operation: ' . $operation);
                        return false;
                }
                $this->debug('opData:');
                $this->appendDebug($this->varDump($opData));

                // Get encoding style for output and set to current
                $encodingStyle = 'http://schemas.xmlsoap.org/soap/encoding/';
                if(($direction == 'input') && isset($opData['output']['encodingStyle']) && ($opData['output']['encodingStyle'] != $encodingStyle)) {
                        $encodingStyle = $opData['output']['encodingStyle'];
```

```
                $enc_style = $encodingStyle;
            }
            // set input params
            $xml = '';
            if (isset($opData[$direction]['parts']) &&
sizeof($opData[$direction]['parts']) > 0) {

                $use = $opData[$direction]['use'];
                $this->debug("use=$use");
                $this->debug('got ' . count($opData[$direction]['parts']) . '
part(s)');
                if (is_array($parameters)) {
                    $parametersArrayType = $this-
>isArraySimpleOrStruct($parameters);
                    $this->debug('have ' . $parametersArrayType . '
parameters');
                    foreach($opData[$direction]['parts'] as $name => $type)
{
                        $this->debug('serializing part "'.$name.'" of type
"'.$type.'"');
                        // Track encoding style
```

```
                                    if(isset($opData[$direction]['encodingStyle'])
&& $encodingStyle != $opData[$direction]['encodingStyle']) {
                                        $encodingStyle =
$opData[$direction]['encodingStyle'];
                                        $enc_style = $encodingStyle;
                                    } else {
                                        $enc_style = false;
                                    }
                                    // NOTE: add error handling here
                                    // if serializeType returns false, then catch global
error and fault
                                    if ($parametersArrayType == 'arraySimple') {
                                        $p = array_shift($parameters);
                                        $this->debug('calling serializeType
w/indexed param');
                                        $xml .= $this->serializeType($name,
$type, $p, $use, $enc_style);
                                    } elseif (isset($parameters[$name])) {
                                        $this->debug('calling serializeType
w/named param');
                                        $xml .= $this->serializeType($name,
$type, $parameters[$name], $use, $enc_style);
```

```
                                        } else {
                                                // TODO: only send nillable
                                                $this->debug('calling serializeType
w/null param');
                                                $xml .= $this->serializeType($name,
$type, null, $use, $enc_style);
                                        }
                                }
                        } else {
                                $this->debug('no parameters passed.');
                        }
                }
                $this->debug("serializeParameters returning: $xml");
                return $xml;
        }


        /**
         * serializes a PHP value according a given type definition
         *
         * @param string $name name of value (part or element)
         * @param string $type XML schema type of value (type or element)
```

```
     * @param mixed $value a native PHP value (parameter value)
     * @param string $use use for part (encoded|literal)
     * @param string $encodingStyle SOAP encoding style for the value (if different than the enclosing style)
     * @param boolean $unqualified a kludge for what should be XML namespace form handling
     * @return string value serialized as an XML string
     * @access private
     */
    function serializeType($name, $type, $value, $use='encoded', $encodingStyle=false, $unqualified=false)
    {
        $this->debug("in serializeType: name=$name, type=$type, use=$use, encodingStyle=$encodingStyle, unqualified=" . ($unqualified ? "unqualified" : "qualified"));
        $this->appendDebug("value=" . $this->varDump($value));
        if($use == 'encoded' && $encodingStyle) {
            $encodingStyle = ' SOAP-ENV:encodingStyle="' . $encodingStyle . '"';
        }
        // if a soapval has been supplied, let its type override the WSDL
    if (is_object($value) && get_class($value) == 'soapval') {
```

```
                if ($value->type_ns) {
                        $type = $value->type_ns . ':' . $value->type;
                        $forceType = true;
                        $this->debug("in serializeType: soapval overrides type to
$type");
                } elseif ($value->type) {
                        $type = $value->type;
                        $forceType = true;
                        $this->debug("in serializeType: soapval overrides type to
$type");
                } else {
                        $forceType = false;
                        $this->debug("in serializeType: soapval does not override
type");
                }
                $attrs = $value->attributes;
                $value = $value->value;
                $this->debug("in serializeType: soapval overrides value to $value");
                if ($attrs) {
                        if (!is_array($value)) {
                                $value['!'] = $value;
```

```
                    }
                    foreach ($attrs as $n => $v) {
                        $value['!' . $n] = $v;
                    }
                    $this->debug("in serializeType: soapval provides attributes");
                }
        } else {
           $forceType = false;
        }
                $xml = '';
                if (strpos($type, ':')) {
                    $uqType = substr($type, strrpos($type, ':') + 1);
                    $ns = substr($type, 0, strrpos($type, ':'));
                    $this->debug("in serializeType: got a prefixed type: $uqType,
$ns");
                    if ($this->getNamespaceFromPrefix($ns)) {
                        $ns = $this->getNamespaceFromPrefix($ns);
                        $this->debug("in serializeType: expanded prefixed type:
$uqType, $ns");
                    }
```

```
                    if($ns == $this->XMLSchemaVersion || $ns ==
'http://schemas.xmlsoap.org/soap/encoding/'){
                        $this->debug('in serializeType: type namespace
indicates XML Schema or SOAP Encoding type');
                        if ($unqualified && $use == 'literal') {
                            $elementNS = " xmlns=\"\"";
                        } else {
                            $elementNS = '';
                        }
                        if (is_null($value)) {
                            if ($use == 'literal') {
                                // TODO: depends on minOccurs
                                $xml = "<$name$elementNS/>";
                            } else {
                                // TODO: depends on nillable, which
should be checked before calling this method
                                $xml = "<$name$elementNS
xsi:nil=\"true\" xsi:type=\"" . $this->getPrefixFromNamespace($ns) . ":$uqType\"/>";
                            }
                            $this->debug("in serializeType: returning:
$xml");
                            return $xml;
```

```
                }
                if ($uqType == 'Array') {
                        // JBoss/Axis does this sometimes
                        return $this->serialize_val($value, $name, false,
false, false, false, $use);
                }
            if ($uqType == 'boolean') {
                if ((is_string($value) && $value == 'false') || (! $value))
{
                        $value = 'false';
                } else {
                        $value = 'true';
                }
                }
                if ($uqType == 'string' && gettype($value) == 'string') {
                        $value = $this->expandEntities($value);
                }
                if (($uqType == 'long' || $uqType == 'unsignedLong')
&& gettype($value) == 'double') {
                        $value = sprintf("%.0lf", $value);
                }
```

```
                    // it's a scalar
                    // TODO: what about null/nil values?
                    // check type isn't a custom type extending xmlschema
namespace
                    if (!$this->getTypeDef($uqType, $ns)) {
                        if ($use == 'literal') {
                            if ($forceType) {
                                $xml = "<$name$elementNS
xsi:type=\"" . $this->getPrefixFromNamespace($ns) .
":$uqType\">$value</$name>";
                            } else {
                                $xml =
"<$name$elementNS>$value</$name>";
                            }
                        } else {
                            $xml = "<$name$elementNS xsi:type=\""
. $this->getPrefixFromNamespace($ns) .
":$uqType\"$encodingStyle>$value</$name>";
                        }
                        $this->debug("in serializeType: returning:
$xml");
                        return $xml;
```

```
                    }
                    $this->debug('custom type extends XML Schema or
SOAP Encoding namespace (yuck)');
                } else if ($ns == 'http://xml.apache.org/xml-soap') {
                    $this->debug('in serializeType: appears to be Apache
SOAP type');
                    if ($uqType == 'Map') {
                        $tt_prefix = $this-
>getPrefixFromNamespace('http://xml.apache.org/xml-soap');
                        if (! $tt_prefix) {
                            $this->debug('in serializeType: Add
namespace for Apache SOAP type');
                            $tt_prefix = 'ns' . rand(1000, 9999);
                            $this->namespaces[$tt_prefix] =
'http://xml.apache.org/xml-soap';
                            // force this to be added to
usedNamespaces
                            $tt_prefix = $this-
>getPrefixFromNamespace('http://xml.apache.org/xml-soap');
                        }
                        $contents = '';
                        foreach($value as $k => $v) {
```

```
                $this->debug("serializing map element:
key $k, value $v");
                $contents .= '<item>';
                $contents .= $this-
>serialize_val($k,'key',false,false,false,false,$use);
                $contents .= $this-
>serialize_val($v,'value',false,false,false,false,$use);
                $contents .= '</item>';
            }
            if ($use == 'literal') {
                if ($forceType) {
                    $xml = "<$name xsi:type=\"" .
$tt_prefix . ":$uqType\">$contents</$name>";
                } else {
                    $xml =
"<$name>$contents</$name>";
                }
            } else {
                $xml = "<$name xsi:type=\"" . $tt_prefix
. ":$uqType\"$encodingStyle>$contents</$name>";
            }
```

```
                                        $this->debug("in serializeType: returning:
$xml");
                                        return $xml;
                                }
                                $this->debug('in serializeType: Apache SOAP type, but
only support Map');
                        }
                } else {
                        // TODO: should the type be compared to types in XSD, and
the namespace
                        // set to XSD if the type matches?
                        $this->debug("in serializeType: No namespace for type
$type");
                        $ns = '';
                        $uqType = $type;
                }
                if(!$typeDef = $this->getTypeDef($uqType, $ns)){
                        $this->setError("$type ($uqType) is not a supported type.");
                        $this->debug("in serializeType: $type ($uqType) is not a
supported type.");
                        return false;
                } else {
```

```
                $this->debug("in serializeType: found typeDef");
                $this->appendDebug('typeDef=' . $this->varDump($typeDef));
                if (substr($uqType, -1) == '^') {
                    $uqType = substr($uqType, 0, -1);
                }
            }
            if (!isset($typeDef['phpType'])) {
                $this->setError("$type ($uqType) has no phpType.");
                $this->debug("in serializeType: $type ($uqType) has no
phpType.");
                return false;
            }
            $phpType = $typeDef['phpType'];
            $this->debug("in serializeType: uqType: $uqType, ns: $ns, phptype:
$phpType, arrayType: " . (isset($typeDef['arrayType']) ? $typeDef['arrayType'] : '') );
            // if php type == struct, map value to the <all> element names
            if ($phpType == 'struct') {
                if (isset($typeDef['typeClass']) && $typeDef['typeClass'] ==
'element') {
                    $elementName = $uqType;
```

```
                    if (isset($typeDef['form']) && ($typeDef['form'] ==
'qualified')) {
                        $elementNS = " xmlns=\"$ns\"";
                    } else {
                        $elementNS = " xmlns=\"\"";
                    }
                } else {
                    $elementName = $name;
                    if ($unqualified) {
                        $elementNS = " xmlns=\"\"";
                    } else {
                        $elementNS = '';
                    }
                }
                if (is_null($value)) {
                    if ($use == 'literal') {
                        // TODO: depends on minOccurs and nillable
                        $xml = "<$elementName$elementNS/>";
                    } else {
                        $xml = "<$elementName$elementNS
xsi:nil=\"true\" xsi:type=\"" . $this->getPrefixFromNamespace($ns) . ":$uqType\"/>";
```

```
                }
                $this->debug("in serializeType: returning: $xml");
                return $xml;
            }
            if (is_object($value)) {
                $value = get_object_vars($value);
            }
            if (is_array($value)) {
                $elementAttrs = $this->serializeComplexTypeAttributes($typeDef, $value, $ns, $uqType);
                if ($use == 'literal') {
                    if ($forceType) {
                        $xml = "<$elementName$elementNS$elementAttrs xsi:type=\"" . $this->getPrefixFromNamespace($ns) . ":$uqType\">";
                    } else {
                        $xml = "<$elementName$elementNS$elementAttrs>";
                    }
                } else {
```

```
                                        $xml =
"<$elementName$elementNS$elementAttrs xsi:type=\"" . $this-
>getPrefixFromNamespace($ns) . ":$uqType\"$encodingStyle>";
                                    }
                                    if (isset($typeDef['simpleContent']) &&
$typeDef['simpleContent'] == 'true') {
                                        if (isset($value['!'])) {
                                            $xml .= $value['!'];
                                            $this->debug("in serializeType:
serialized simpleContent for type $type");
                                        } else {
                                            $this->debug("in serializeType: no
simpleContent to serialize for type $type");
                                        }
                                    } else {
                                        // complexContent
                                        $xml .= $this-
>serializeComplexTypeElements($typeDef, $value, $ns, $uqType, $use,
$encodingStyle);
                                    }
                                    $xml .= "</$elementName>";
                                } else {
```

```
                    $this->debug("in serializeType: phpType is struct, but
value is not an array");
                    $this->setError("phpType is struct, but value is not an
array: see debug output for details");
                    $xml = '';
                }
            } elseif ($phpType == 'array') {
                if (isset($typeDef['form']) && ($typeDef['form'] == 'qualified'))
{
                    $elementNS = " xmlns=\"$ns\"";
                } else {
                    if ($unqualified) {
                        $elementNS = " xmlns=\"\"";
                    } else {
                        $elementNS = '';
                    }
                }
                if (is_null($value)) {
                    if ($use == 'literal') {
                        // TODO: depends on minOccurs
                        $xml = "<$name$elementNS/>";
```

```
                } else {
                        $xml = "<$name$elementNS xsi:nil=\"true\"
xsi:type=\"" .
                                $this-
>getPrefixFromNamespace('http://schemas.xmlsoap.org/soap/encoding/') .
                                ":Array\" " .
                                $this-
>getPrefixFromNamespace('http://schemas.xmlsoap.org/soap/encoding/') .
                                ':arrayType="' .
                                $this->getPrefixFromNamespace($this-
>getPrefix($typeDef['arrayType'])) .
                                ':' .
                                $this-
>getLocalPart($typeDef['arrayType'])."[0]\"/>";
                }
                $this->debug("in serializeType: returning: $xml");
                return $xml;
        }
        if (isset($typeDef['multidimensional'])) {
                $nv = array();
                foreach($value as $v) {
                        $cols = ',' . sizeof($v);
```

```
                    $nv = array_merge($nv, $v);
                }
                $value = $nv;
            } else {
                $cols = '';
            }
            if (is_array($value) && sizeof($value) >= 1) {
                $rows = sizeof($value);
                $contents = '';
                foreach($value as $k => $v) {
                    $this->debug("serializing array element: $k, $v
of type: $typeDef[arrayType]");
                    //if (strpos($typeDef['arrayType'], ':') ) {
                    if (!in_array($typeDef['arrayType'],$this-
>typemap['http://www.w3.org/2001/XMLSchema'])) {
                        $contents .= $this->serializeType('item',
$typeDef['arrayType'], $v, $use);
                    } else {
                        $contents .= $this->serialize_val($v, 'item',
$typeDef['arrayType'], null, $this->XMLSchemaVersion, false, $use);
                    }
```

```
                }
            } else {
                $rows = 0;
                $contents = null;
            }
            // TODO: for now, an empty value will be serialized as a zero element
            // array.  Revisit this when coding the handling of null/nil values.
            if ($use == 'literal') {
                $xml = "<$name$elementNS>"
                    .$contents
                    ."</$name>";
            } else {
                $xml = "<$name$elementNS xsi:type=\"".$this->getPrefixFromNamespace('http://schemas.xmlsoap.org/soap/encoding/').':Array" '.
                    $this->getPrefixFromNamespace('http://schemas.xmlsoap.org/soap/encoding/')
                    .':arrayType="'
                    .$this->getPrefixFromNamespace($this->getPrefix($typeDef['arrayType']))
```

```
                                        .":".$this-
>getLocalPart($typeDef['arrayType'])."[$rows$cols]\">"
                                        .$contents
                                        ."</$name>";
                        }
                } elseif ($phpType == 'scalar') {
                        if (isset($typeDef['form']) && ($typeDef['form'] == 'qualified'))
{
                                $elementNS = " xmlns=\"$ns\"";
                        } else {
                                if ($unqualified) {
                                        $elementNS = " xmlns=\"\"";
                                } else {
                                        $elementNS = '';
                                }
                        }
                        if ($use == 'literal') {
                                if ($forceType) {
                                        $xml = "<$name$elementNS xsi:type=\"" .
$this->getPrefixFromNamespace($ns) . ":$uqType\">$value</$name>";
                                } else {
```

```
                                        $xml = "<$name$elementNS>$value</$name>";
                                }
                        } else {
                                $xml = "<$name$elementNS xsi:type=\"" . $this->getPrefixFromNamespace($ns) . ":$uqType\"$encodingStyle>$value</$name>";
                        }
                }
                $this->debug("in serializeType: returning: $xml");
                return $xml;
        }
        /**
         * serializes the attributes for a complexType
         *
         * @param array $typeDef our internal representation of an XML schema type (or element)
         * @param mixed $value a native PHP value (parameter value)
         * @param string $ns the namespace of the type
         * @param string $uqType the local part of the type
         * @return string value serialized as an XML string
         * @access private
         */
```

```
        function serializeComplexTypeAttributes($typeDef, $value, $ns, $uqType) {
                $this->debug("serializeComplexTypeAttributes for XML Schema type $ns:$uqType");
                $xml = '';
                if (isset($typeDef['extensionBase'])) {
                        $nsx = $this->getPrefix($typeDef['extensionBase']);
                        $uqTypex = $this->getLocalPart($typeDef['extensionBase']);
                        if ($this->getNamespaceFromPrefix($nsx)) {
                                $nsx = $this->getNamespaceFromPrefix($nsx);
                        }
                        if ($typeDefx = $this->getTypeDef($uqTypex, $nsx)) {
                                $this->debug("serialize attributes for extension base $nsx:$uqTypex");
                                $xml .= $this->serializeComplexTypeAttributes($typeDefx, $value, $nsx, $uqTypex);
                        } else {
                                $this->debug("extension base $nsx:$uqTypex is not a supported type");
                        }
                }
                if (isset($typeDef['attrs']) && is_array($typeDef['attrs'])) {
```

```
                $this->debug("serialize attributes for XML Schema type
$ns:$uqType");
                if (is_array($value)) {
                        $xvalue = $value;
                } elseif (is_object($value)) {
                        $xvalue = get_object_vars($value);
                } else {
                        $this->debug("value is neither an array nor an object for
XML Schema type $ns:$uqType");
                        $xvalue = array();
                }
                foreach ($typeDef['attrs'] as $aName => $attrs) {
                        if (isset($xvalue['!' . $aName])) {
                                $xname = '!' . $aName;
                                $this->debug("value provided for attribute
$aName with key $xname");
                        } elseif (isset($xvalue[$aName])) {
                                $xname = $aName;
                                $this->debug("value provided for attribute
$aName with key $xname");
                        } elseif (isset($attrs['default'])) {
```

```
                                        $xname = '!' . $aName;
                                        $xvalue[$xname] = $attrs['default'];
                                        $this->debug('use default value of ' .
$xvalue[$aName] . ' for attribute ' . $aName);
                                    } else {
                                        $xname = '';
                                        $this->debug("no value provided for attribute
$aName");
                                    }
                                    if ($xname) {
                                        $xml .=  " $aName=\"" . $this-
>expandEntities($xvalue[$xname]) . "\"";
                                    }
                                }
                            } else {
                                $this->debug("no attributes to serialize for XML Schema type
$ns:$uqType");
                            }
                            return $xml;
                        }
```

```
    /**
     * serializes the elements for a complexType
     *
     * @param array $typeDef our internal representation of an XML schema type (or element)
     * @param mixed $value a native PHP value (parameter value)
     * @param string $ns the namespace of the type
     * @param string $uqType the local part of the type
     * @param string $use use for part (encoded|literal)
     * @param string $encodingStyle SOAP encoding style for the value (if different than the enclosing style)
     * @return string value serialized as an XML string
     * @access private
     */
    function serializeComplexTypeElements($typeDef, $value, $ns, $uqType, $use='encoded', $encodingStyle=false) {
        $this->debug("in serializeComplexTypeElements for XML Schema type $ns:$uqType");
        $xml = '';
        if (isset($typeDef['extensionBase'])) {
            $nsx = $this->getPrefix($typeDef['extensionBase']);
```

```
                $uqTypex = $this->getLocalPart($typeDef['extensionBase']);
                if ($this->getNamespaceFromPrefix($nsx)) {
                        $nsx = $this->getNamespaceFromPrefix($nsx);
                }
                if ($typeDefx = $this->getTypeDef($uqTypex, $nsx)) {
                        $this->debug("serialize elements for extension base $nsx:$uqTypex");
                        $xml .= $this->serializeComplexTypeElements($typeDefx, $value, $nsx, $uqTypex, $use, $encodingStyle);
                } else {
                        $this->debug("extension base $nsx:$uqTypex is not a supported type");
                }
        }
        if (isset($typeDef['elements']) && is_array($typeDef['elements'])) {
                $this->debug("in serializeComplexTypeElements, serialize elements for XML Schema type $ns:$uqType");
                if (is_array($value)) {
                        $xvalue = $value;
                } elseif (is_object($value)) {
                        $xvalue = get_object_vars($value);
```

```
                } else {
                        $this->debug("value is neither an array nor an object for
XML Schema type $ns:$uqType");
                        $xvalue = array();
                }
                // toggle whether all elements are present - ideally should
validate against schema
                if (count($typeDef['elements']) != count($xvalue)){
                        $optionals = true;
                }
                foreach ($typeDef['elements'] as $eName => $attrs) {
                        if (!isset($xvalue[$eName])) {
                                if (isset($attrs['default'])) {
                                        $xvalue[$eName] = $attrs['default'];
                                        $this->debug('use default value of ' .
$xvalue[$eName] . ' for element ' . $eName);
                                }
                        }
                        // if user took advantage of a minOccurs=0, then only
serialize named parameters
                        if (isset($optionals)
```

```
                                && (!isset($xvalue[$eName]))
                                    && ( (!isset($attrs['nillable'])) || $attrs['nillable'] != 'true')
                                    ){
                                    if (isset($attrs['minOccurs']) && $attrs['minOccurs'] <> '0') {
                                        $this->debug("apparent error: no value provided for element $eName with minOccurs=" . $attrs['minOccurs']);
                                    }
                                    // do nothing
                                    $this->debug("no value provided for complexType element $eName and element is not nillable, so serialize nothing");
                                } else {
                                    // get value
                                    if (isset($xvalue[$eName])) {
                                       $v = $xvalue[$eName];
                                    } else {
                                       $v = null;
                                    }
                                    if (isset($attrs['form'])) {
                                        $unqualified = ($attrs['form'] == 'unqualified');
```

```
                } else {
                    $unqualified = false;
                }
                if (isset($attrs['maxOccurs']) &&
($attrs['maxOccurs'] == 'unbounded' || $attrs['maxOccurs'] > 1) && isset($v) &&
is_array($v) && $this->isArraySimpleOrStruct($v) == 'arraySimple') {
                    $vv = $v;
                    foreach ($vv as $k => $v) {
                        if (isset($attrs['type']) ||
isset($attrs['ref'])) {
                            // serialize schema-defined
type
                            $xml .= $this-
>serializeType($eName, isset($attrs['type']) ? $attrs['type'] : $attrs['ref'], $v, $use,
$encodingStyle, $unqualified);
                        } else {
                            // serialize generic type
(can this ever really happen?)
                            $this->debug("calling
serialize_val() for $v, $eName, false, false, false, false, $use");
                            $xml .= $this-
>serialize_val($v, $eName, false, false, false, false, $use);
                        }
```

```
                }
            } else {
                if (is_null($v) &&
isset($attrs['minOccurs']) && $attrs['minOccurs'] == '0') {
                    // do nothing
                } elseif (is_null($v) &&
isset($attrs['nillable']) && $attrs['nillable'] == 'true') {
                    // TODO: serialize a nil correctly,
but for now serialize schema-defined type
                    $xml .= $this->serializeType($eName,
isset($attrs['type']) ? $attrs['type'] : $attrs['ref'], $v, $use, $encodingStyle,
$unqualified);
                } elseif (isset($attrs['type']) ||
isset($attrs['ref'])) {
                    // serialize schema-defined type
                    $xml .= $this->serializeType($eName,
isset($attrs['type']) ? $attrs['type'] : $attrs['ref'], $v, $use, $encodingStyle,
$unqualified);
                } else {
                    // serialize generic type (can this
ever really happen?)
                    $this->debug("calling serialize_val()
for $v, $eName, false, false, false, false, $use");
```

```
                                        $xml .= $this->serialize_val($v,
$eName, false, false, false, false, $use);
                                    }
                                }
                            }
                        }
                    } else {
                        $this->debug("no elements to serialize for XML Schema type
$ns:$uqType");
                    }
                    return $xml;
                }
                /**
                * adds an XML Schema complex type to the WSDL types
                *
                * @param string         $name
                * @param string $typeClass (complexType|simpleType|attribute)
                * @param string $phpType currently supported are array and struct (php assoc
array)
                * @param string $compositor (all|sequence|choice)
```

```
        * @param string $restrictionBase namespace:name
(http://schemas.xmlsoap.org/soap/encoding/:Array)

        * @param array $elements e.g. array ( name => array(name=>'',type=>'') )

        * @param array $attrs e.g. array(array('ref'=>'SOAP-
ENC:arrayType','wsdl:arrayType'=>'xsd:string[]'))

        * @param string $arrayType as namespace:name (xsd:string)

        * @see nusoap_xmlschema

        * @access public

        */

        function
addComplexType($name,$typeClass='complexType',$phpType='array',$compositor=
'',$restrictionBase='',$elements=array(),$attrs=array(),$arrayType='') {

                if (count($elements) > 0) {

                        $eElements = array();

                foreach($elements as $n => $e){

                // expand each element

                $ee = array();

                foreach ($e as $k => $v) {

                        $k = strpos($k,':') ? $this->expandQname($k) : $k;

                        $v = strpos($v,':') ? $this->expandQname($v) : $v;

                        $ee[$k] = $v;
```

```
                }
                $eElements[$n] = $ee;
        }
        $elements = $eElements;
        }
        if (count($attrs) > 0) {
        foreach($attrs as $n => $a){
        // expand each attribute
        foreach ($a as $k => $v) {
                $k = strpos($k,':') ? $this->expandQname($k) : $k;
                $v = strpos($v,':') ? $this->expandQname($v) : $v;
                $aa[$k] = $v;
                }
                $eAttrs[$n] = $aa;
        }
        $attrs = $eAttrs;
        }

        $restrictionBase = strpos($restrictionBase,':') ? $this-
>expandQname($restrictionBase) : $restrictionBase;
```

```
                $arrayType = strpos($arrayType,':') ? $this-
>expandQname($arrayType) : $arrayType;

                $typens = isset($this->namespaces['types']) ? $this-
>namespaces['types'] : $this->namespaces['tns'];

                $this->schemas[$typens][0]-
>addComplexType($name,$typeClass,$phpType,$compositor,$restrictionBase,$elem
ents,$attrs,$arrayType);

        }

        /**

        * adds an XML Schema simple type to the WSDL types

        *

        * @param string $name

        * @param string $restrictionBase namespace:name
(http://schemas.xmlsoap.org/soap/encoding/:Array)

        * @param string $typeClass (should always be simpleType)

        * @param string $phpType (should always be scalar)

        * @param array $enumeration array of values

        * @see nusoap_xmlschema

        * @access public

        */

        function addSimpleType($name, $restrictionBase='',
$typeClass='simpleType', $phpType='scalar', $enumeration=array()) {
```

```
            $restrictionBase = strpos($restrictionBase,':') ? $this->expandQname($restrictionBase) : $restrictionBase;

            $typens = isset($this->namespaces['types']) ? $this->namespaces['types'] : $this->namespaces['tns'];

            $this->schemas[$typens][0]->addSimpleType($name, $restrictionBase, $typeClass, $phpType, $enumeration);

        }

        /**

        * adds an element to the WSDL types

        *

        * @param array $attrs attributes that must include name and type

        * @see nusoap_xmlschema

        * @access public

        */

        function addElement($attrs) {

            $typens = isset($this->namespaces['types']) ? $this->namespaces['types'] : $this->namespaces['tns'];

            $this->schemas[$typens][0]->addElement($attrs);

        }


        /**
```

```
    * register an operation with the server
    *
    * @param string $name operation (method) name
    * @param array $in assoc array of input values: key = param name, value = param type
    * @param array $out assoc array of output values: key = param name, value = param type
    * @param string $namespace optional The namespace for the operation
    * @param string $soapaction optional The soapaction for the operation
    * @param string $style (rpc|document) optional The style for the operation Note: when 'document' is specified, parameter and return wrappers are created for you automatically
    * @param string $use (encoded|literal) optional The use for the parameters (cannot mix right now)
    * @param string $documentation optional The description to include in the WSDL
    * @param string $encodingStyle optional (usually 'http://schemas.xmlsoap.org/soap/encoding/' for encoded)
    * @access public
    */
    function addOperation($name, $in = false, $out = false, $namespace = false, $soapaction = false, $style = 'rpc', $use = 'encoded', $documentation = '', $encodingStyle = ''){
```

```
        if ($use == 'encoded' && $encodingStyle == '') {
            $encodingStyle = 'http://schemas.xmlsoap.org/soap/encoding/';
        }

        if ($style == 'document') {
            $elements = array();
            foreach ($in as $n => $t) {
                $elements[$n] = array('name' => $n, 'type' => $t, 'form'
=> 'unqualified');
            }
            $this->addComplexType($name . 'RequestType',
'complexType', 'struct', 'all', '', $elements);
            $this->addElement(array('name' => $name, 'type' => $name .
'RequestType'));
            $in = array('parameters' => 'tns:' . $name . '^');

            $elements = array();
            foreach ($out as $n => $t) {
                $elements[$n] = array('name' => $n, 'type' => $t, 'form'
=> 'unqualified');
            }
```

```
                $this->addComplexType($name . 'ResponseType',
'complexType', 'struct', 'all', '', $elements);

                $this->addElement(array('name' => $name . 'Response', 'type'
=> $name . 'ResponseType', 'form' => 'qualified'));

                $out = array('parameters' => 'tns:' . $name . 'Response' . '^');

        }

        // get binding

        $this->bindings[ $this->serviceName . 'Binding' ]['operations'][$name]
=

        array(

        'name' => $name,

        'binding' => $this->serviceName . 'Binding',

        'endpoint' => $this->endpoint,

        'soapAction' => $soapaction,

        'style' => $style,

        'input' => array(

                'use' => $use,

                'namespace' => $namespace,

                'encodingStyle' => $encodingStyle,

                'message' => $name . 'Request',

                'parts' => $in),
```

```
                'output' => array(
                        'use' => $use,
                        'namespace' => $namespace,
                        'encodingStyle' => $encodingStyle,
                        'message' => $name . 'Response',
                        'parts' => $out),
                'namespace' => $namespace,
                'transport' => 'http://schemas.xmlsoap.org/soap/http',
                'documentation' => $documentation);
                // add portTypes
                // add messages
                if($in)
                {
                        foreach($in as $pName => $pType)
                        {
                                if(strpos($pType,':')) {
                                        $pType = $this-
>getNamespaceFromPrefix($this->getPrefix($pType)).":".$this-
>getLocalPart($pType);
                                }
                                $this->messages[$name.'Request'][$pName] = $pType;
```

```
                }
            } else {
        $this->messages[$name.'Request']= '0';
    }
            if($out)
            {
                foreach($out as $pName => $pType)
                {
                    if(strpos($pType,':')) {
                        $pType = $this->getNamespaceFromPrefix($this->getPrefix($pType)).":".$this->getLocalPart($pType);
                    }
                    $this->messages[$name.'Response'][$pName] = $pType;
                }
            } else {
        $this->messages[$name.'Response']= '0';
    }
            return true;
        }
```

```
}
?><?php
/**
*
* nusoap_parser class parses SOAP XML messages into native PHP values
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class nusoap_parser extends nusoap_base {
        var $xml = '';
        var $xml_encoding = '';
        var $method = '';
        var $root_struct = '';
        var $root_struct_name = '';
        var $root_struct_namespace = '';
        var $root_header = '';
    var $document = '';                 // incoming SOAP body (text)
```

```
        // determines where in the message we are (envelope,header,body,method)
        var $status = '';
        var $position = 0;
        var $depth = 0;
        var $default_namespace = '';
        var $namespaces = array();
        var $message = array();
    var $parent = '';
        var $fault = false;
        var $fault_code = '';
        var $fault_str = '';
        var $fault_detail = '';
        var $depth_array = array();
        var $debug_flag = true;
        var $soapresponse = NULL;  // parsed SOAP Body
        var $soapheader = NULL;                // parsed SOAP Header
        var $responseHeaders = '';     // incoming SOAP headers (text)
        var $body_position = 0;
        // for multiref parsing:
        // array of id => pos
```

```
        var $ids = array();
        // array of id => hrefs => pos
        var $multirefs = array();
        // toggle for auto-decoding element content
        var $decode_utf8 = true;


        /**
        * constructor that actually does the parsing
        *
        * @param    string $xml SOAP message
        * @param    string $encoding character encoding scheme of message
        * @param    string $method method for which XML is parsed (unused?)
        * @param    string $decode_utf8 whether to decode UTF-8 to ISO-8859-1
        * @access   public
        */
        function nusoap_parser($xml,$encoding='UTF-
8',$method='',$decode_utf8=true){
                parent::nusoap_base();
                $this->xml = $xml;
                $this->xml_encoding = $encoding;
```

```
                $this->method = $method;
                $this->decode_utf8 = $decode_utf8;
                // Check whether content has been read.
                if(!empty($xml)){
                        // Check XML encoding
                        $pos_xml = strpos($xml, '<?xml');
                        if ($pos_xml !== FALSE) {
                                $xml_decl = substr($xml, $pos_xml, strpos($xml, '?>',
$pos_xml + 2) - $pos_xml + 1);
                                if (preg_match("/encoding=[\"'](.[^\"']*)[\"']/",
$xml_decl, $res)) {
                                        $xml_encoding = $res[1];
                                        if (strtoupper($xml_encoding) != $encoding) {
                                                $err = "Charset from HTTP Content-
Type '" . $encoding . "' does not match encoding from XML declaration '" .
$xml_encoding . "'";
                                                $this->debug($err);
                                                if ($encoding != 'ISO-8859-1' ||
strtoupper($xml_encoding) != 'UTF-8') {
                                                        $this->setError($err);
                                                        return;
                                                }
```

```
                                        // when HTTP says ISO-8859-1 (the
default) and XML says UTF-8 (the typical), assume the other endpoint is just sloppy
and proceed
                                    } else {
                                        $this->debug('Charset from HTTP
Content-Type matches encoding from XML declaration');
                                    }
                                } else {
                                    $this->debug('No encoding specified in XML
declaration');
                                }
                            } else {
                                $this->debug('No XML declaration');
                            }
                            $this->debug('Entering nusoap_parser(), length='.strlen($xml).',
encoding='.$encoding);
                            // Create an XML parser - why not xml_parser_create_ns?
                            $this->parser = xml_parser_create($this->xml_encoding);
                            // Set the options for parsing the XML data.
                            //xml_parser_set_option($parser,
XML_OPTION_SKIP_WHITE, 1);
```

```
                    xml_parser_set_option($this->parser,
XML_OPTION_CASE_FOLDING, 0);
                    xml_parser_set_option($this->parser,
XML_OPTION_TARGET_ENCODING, $this->xml_encoding);
                    // Set the object for the parser.
                    xml_set_object($this->parser, $this);
                    // Set the element handlers for the parser.
                    xml_set_element_handler($this->parser,
'start_element','end_element');
                    xml_set_character_data_handler($this-
>parser,'character_data');
                    // Parse the XML file.
                    if(!xml_parse($this->parser,$xml,true)){
                       // Display an error message.
                       $err = sprintf('XML error parsing SOAP payload on line %d:
%s',
                       xml_get_current_line_number($this->parser),
                       xml_error_string(xml_get_error_code($this->parser)));
                            $this->debug($err);
                            $this->debug("XML payload:\n" . $xml);
                            $this->setError($err);
                    } else {
```

```
                $this->debug('in nusoap_parser ctor, message:');
                $this->appendDebug($this->varDump($this->message));
                $this->debug('parsed successfully, found root struct: '.$this->root_struct.' of name '.$this->root_struct_name);
                // get final value
                $this->soapresponse = $this->message[$this->root_struct]['result'];
                // get header value
                if($this->root_header != '' && isset($this->message[$this->root_header]['result'])){
                    $this->soapheader = $this->message[$this->root_header]['result'];
                }
                // resolve hrefs/ids
                if(sizeof($this->multirefs) > 0){
                    foreach($this->multirefs as $id => $hrefs){
                        $this->debug('resolving multirefs for id: '.$id);
                        $idVal = $this->buildVal($this->ids[$id]);
                        if (is_array($idVal) && isset($idVal['!id'])) {
```

```
                                        unset($idVal['!id']);
                                    }
                                    foreach($hrefs as $refPos => $ref){
                                        $this->debug('resolving href at
pos '.$refPos);
                                        $this->multirefs[$id][$refPos] =
$idVal;
                                    }
                                }
                            }
                        }
                        xml_parser_free($this->parser);
                    } else {
                        $this->debug('xml was empty, didn\'t parse!');
                        $this->setError('xml was empty, didn\'t parse!');
                    }
                }

                /**
                * start-element handler
                *
```

```
* @param    resource $parser XML parser object
* @param    string $name element name
* @param    array $attrs associative array of attributes
* @access   private
*/
function start_element($parser, $name, $attrs) {
        // position in a total number of elements, starting from 0
        // update class level pos
        $pos = $this->position++;
        // and set mine
        $this->message[$pos] = array('pos' => $pos,'children'=>'','cdata'=>'');
        // depth = how many levels removed from root?
        // set mine as current global depth and increment global depth value
        $this->message[$pos]['depth'] = $this->depth++;

        // else add self as child to whoever the current parent is
        if($pos != 0){
                $this->message[$this->parent]['children'] .= '|'.$pos;
        }
        // set my parent
```

```
$this->message[$pos]['parent'] = $this->parent;
// set self as current parent
$this->parent = $pos;
// set self as current value for this depth
$this->depth_array[$this->depth] = $pos;
// get element prefix
if(strpos($name,':')){
        // get ns prefix
        $prefix = substr($name,0,strpos($name,':'));
        // get unqualified name
        $name = substr(strstr($name,':'),1);
}
// set status
if ($name == 'Envelope' && $this->status == '') {
        $this->status = 'envelope';
} elseif ($name == 'Header' && $this->status == 'envelope') {
        $this->root_header = $pos;
        $this->status = 'header';
} elseif ($name == 'Body' && $this->status == 'envelope'){
        $this->status = 'body';
```

```
                    $this->body_position = $pos;
                // set method
                } elseif($this->status == 'body' && $pos == ($this->body_position+1))
{
                    $this->status = 'method';
                    $this->root_struct_name = $name;
                    $this->root_struct = $pos;
                    $this->message[$pos]['type'] = 'struct';
                    $this->debug("found root struct $this->root_struct_name, pos
$this->root_struct");
                }
                // set my status
                $this->message[$pos]['status'] = $this->status;
                // set name
                $this->message[$pos]['name'] = htmlspecialchars($name);
                // set attrs
                $this->message[$pos]['attrs'] = $attrs;

                // loop through atts, logging ns and type declarations
        $attstr = '';
                foreach($attrs as $key => $value){
```

```
        $key_prefix = $this->getPrefix($key);
                        $key_localpart = $this->getLocalPart($key);
                        // if ns declarations, add to class level array of valid
namespaces
        if($key_prefix == 'xmlns'){
                                if(preg_match('/^http:\/\/www.w3.org\/[0-
9]{4}\/XMLSchema$/',$value)){
                                        $this->XMLSchemaVersion = $value;
                                        $this->namespaces['xsd'] = $this-
>XMLSchemaVersion;
                                        $this->namespaces['xsi'] = $this-
>XMLSchemaVersion.'-instance';
                                }
            $this->namespaces[$key_localpart] = $value;
                                // set method namespace
                                if($name == $this->root_struct_name){
                                        $this->methodNamespace = $value;
                                }
                        // if it's a type declaration, set type
      } elseif($key_localpart == 'type'){
                if (isset($this->message[$pos]['type']) && $this-
>message[$pos]['type'] == 'array') {
```

```
                    // do nothing: already processed arrayType
                } else {
                    $value_prefix = $this->getPrefix($value);
                $value_localpart = $this->getLocalPart($value);
                                $this->message[$pos]['type'] = $value_localpart;
                                $this->message[$pos]['typePrefix'] =
$value_prefix;
                if(isset($this->namespaces[$value_prefix])){
                    $this->message[$pos]['type_namespace'] = $this-
>namespaces[$value_prefix];
                } else if(isset($attrs['xmlns:'.$value_prefix])) {
                                    $this->message[$pos]['type_namespace']
= $attrs['xmlns:'.$value_prefix];
                }
                                // should do something here with the namespace
of specified type?
                            }
                    } elseif($key_localpart == 'arrayType'){
                        $this->message[$pos]['type'] = 'array';
                        /* do arrayType ereg here
                        [1]    arrayTypeValue    ::=    atype asize
```

```
                    [2]    atype    ::=    QName rank*

                    [3]    rank    ::=    '[' (',')* ']'

                    [4]    asize    ::=    '[' length~ ']'

                    [5]    length    ::=    nextDimension* Digit+

                    [6]    nextDimension    ::=    Digit+ ','

                    */

                    $expr = '/([A-Za-z0-9_]+):([A-Za-z]+[A-Za-z0-
9_]+)\[([0-9]+),?([0-9]*)\]/';

                    if(preg_match($expr,$value,$regs)){

                        $this->message[$pos]['typePrefix'] = $regs[1];

                        $this->message[$pos]['arrayTypePrefix'] =
$regs[1];

            if (isset($this->namespaces[$regs[1]])) {

                $this->message[$pos]['arrayTypeNamespace'] = $this-
>namespaces[$regs[1]];

            } else if (isset($attrs['xmlns:'.$regs[1]])) {

                                $this-
>message[$pos]['arrayTypeNamespace'] = $attrs['xmlns:'.$regs[1]];

            }

                        $this->message[$pos]['arrayType'] = $regs[2];

                        $this->message[$pos]['arraySize'] = $regs[3];
```

```
                    $this->message[$pos]['arrayCols'] = $regs[4];
                }
            // specifies nil value (or not)
            } elseif ($key_localpart == 'nil'){
                $this->message[$pos]['nil'] = ($value == 'true' || $value
== '1');
            // some other attribute
            } elseif ($key != 'href' && $key != 'xmlns' && $key_localpart
!= 'encodingStyle' && $key_localpart != 'root') {
                $this->message[$pos]['xattrs']['!' . $key] = $value;
            }
            if ($key == 'xmlns') {
                $this->default_namespace = $value;
            }
            // log id
            if($key == 'id'){
                $this->ids[$value] = $pos;
            }
            // root
            if($key_localpart == 'root' && $value == 1){
                $this->status = 'method';
```

```
                                $this->root_struct_name = $name;
                                $this->root_struct = $pos;
                                $this->debug("found root struct $this->root_struct_name, pos $pos");
                        }
        // for doclit
        $attstr .= " $key=\"$value\"";
                }
      // get namespace - must be done after namespace atts are processed
                if(isset($prefix)){
                        $this->message[$pos]['namespace'] = $this->namespaces[$prefix];
                        $this->default_namespace = $this->namespaces[$prefix];
                } else {
                        $this->message[$pos]['namespace'] = $this->default_namespace;
                }
      if($this->status == 'header'){
        if ($this->root_header != $pos) {
                $this->responseHeaders .= "<" . (isset($prefix) ? $prefix . ':' : '') . "$name$attstr>";
```

```
            }
        } elseif($this->root_struct_name != ''){
            $this->document .= "<" . (isset($prefix) ? $prefix . ':' : '') . "$name$attstr>";
        }
            }

            /**
            * end-element handler
            *
            * @param    resource $parser XML parser object
            * @param    string $name element name
            * @access   private
            */
            function end_element($parser, $name) {
                    // position of current element is equal to the last value left in
depth_array for my depth
                    $pos = $this->depth_array[$this->depth--];

        // get element prefix
                    if(strpos($name,':')){
```

```
                    // get ns prefix
                    $prefix = substr($name,0,strpos($name,':'));
                    // get unqualified name
                    $name = substr(strstr($name,':'),1);
                }

                // build to native type
                if(isset($this->body_position) && $pos > $this->body_position){
                    // deal w/ multirefs
                    if(isset($this->message[$pos]['attrs']['href'])){
                        // get id
                        $id = substr($this->message[$pos]['attrs']['href'],1);
                        // add placeholder to href array
                        $this->multirefs[$id][$pos] = 'placeholder';
                        // add set a reference to it as the result value
                        $this->message[$pos]['result'] =& $this-
>multirefs[$id][$pos];
            // build complexType values
                    } elseif($this->message[$pos]['children'] != ''){
                        // if result has already been generated (struct/array)
```

```
                    if(!isset($this->message[$pos]['result'])){
                        $this->message[$pos]['result'] = $this->buildVal($pos);
                    }
                // build complexType values of attributes and possibly simpleContent
                } elseif (isset($this->message[$pos]['xattrs'])) {
                    if (isset($this->message[$pos]['nil']) && $this->message[$pos]['nil']) {
                        $this->message[$pos]['xattrs']['!'] = null;
                    } elseif (isset($this->message[$pos]['cdata']) && trim($this->message[$pos]['cdata']) != '') {
                if (isset($this->message[$pos]['type'])) {
                            $this->message[$pos]['xattrs']['!'] = $this->decodeSimple($this->message[$pos]['cdata'], $this->message[$pos]['type'], isset($this->message[$pos]['type_namespace']) ? $this->message[$pos]['type_namespace'] : '');
                        } else {
                            $parent = $this->message[$pos]['parent'];
                            if (isset($this->message[$parent]['type']) && ($this->message[$parent]['type'] == 'array') && isset($this->message[$parent]['arrayType'])) {
```

```
                                                $this->message[$pos]['xattrs']['!']
= $this->decodeSimple($this->message[$pos]['cdata'], $this-
>message[$parent]['arrayType'], isset($this-
>message[$parent]['arrayTypeNamespace']) ? $this-
>message[$parent]['arrayTypeNamespace'] : '');

                                        } else {

                                                $this->message[$pos]['xattrs']['!']
= $this->message[$pos]['cdata'];

                                        }

                                }

                        }

                        $this->message[$pos]['result'] = $this-
>message[$pos]['xattrs'];

                // set value of simpleType (or nil complexType)

                } else {

            //$this->debug('adding data for scalar value '.$this-
>message[$pos]['name'].' of value '.$this->message[$pos]['cdata']);

                        if (isset($this->message[$pos]['nil']) && $this-
>message[$pos]['nil']) {

                                $this->message[$pos]['xattrs']['!'] = null;

                        } elseif (isset($this->message[$pos]['type'])) {

                                $this->message[$pos]['result'] = $this-
>decodeSimple($this->message[$pos]['cdata'], $this->message[$pos]['type'],
```

```
isset($this->message[$pos]['type_namespace']) ? $this->message[$pos]['type_namespace'] : '');
                } else {
                    $parent = $this->message[$pos]['parent'];
                    if (isset($this->message[$parent]['type']) && ($this->message[$parent]['type'] == 'array') && isset($this->message[$parent]['arrayType'])) {
                        $this->message[$pos]['result'] = $this->decodeSimple($this->message[$pos]['cdata'], $this->message[$parent]['arrayType'], isset($this->message[$parent]['arrayTypeNamespace']) ? $this->message[$parent]['arrayTypeNamespace'] : '');
                    } else {
                        $this->message[$pos]['result'] = $this->message[$pos]['cdata'];
                    }
                }

                /* add value to parent's result, if parent is struct/array
                $parent = $this->message[$pos]['parent'];
                if($this->message[$parent]['type'] != 'map'){
                    if(strtolower($this->message[$parent]['type']) == 'array'){
```

```
                                        $this->message[$parent]['result'][] =
$this->message[$pos]['result'];
                                    } else {
                                        $this->message[$parent]['result'][$this-
>message[$pos]['name']] = $this->message[$pos]['result'];
                                    }
                                }
                                */
                            }
                        }

    // for doclit
    if($this->status == 'header'){
        if ($this->root_header != $pos) {
            $this->responseHeaders .= "</" . (isset($prefix) ? $prefix . ':' : '') .
"$name>";
        }
    } elseif($pos >= $this->root_struct){
        $this->document .= "</" . (isset($prefix) ? $prefix . ':' : '') . "$name>";
    }
            // switch status
```

```
                if ($pos == $this->root_struct){
                        $this->status = 'body';
                        $this->root_struct_namespace = $this->message[$pos]['namespace'];
                } elseif ($pos == $this->root_header) {
                        $this->status = 'envelope';
                } elseif ($name == 'Body' && $this->status == 'body') {
                        $this->status = 'envelope';
                } elseif ($name == 'Header' && $this->status == 'header') { // will never happen
                        $this->status = 'envelope';
                } elseif ($name == 'Envelope' && $this->status == 'envelope') {
                        $this->status = '';
                }
                // set parent back to my parent
                $this->parent = $this->message[$pos]['parent'];
        }

        /**
        * element content handler
        *
```

```
        * @param    resource $parser XML parser object
        * @param    string $data element content
        * @access   private
        */
        function character_data($parser, $data){
                $pos = $this->depth_array[$this->depth];
                if ($this->xml_encoding=='UTF-8'){
                        // TODO: add an option to disable this for folks who want
                        // raw UTF-8 that, e.g., might not map to iso-8859-1
                        // TODO: this can also be handled with
xml_parser_set_option($this->parser, XML_OPTION_TARGET_ENCODING,
"ISO-8859-1");
                        if($this->decode_utf8){
                                $data = utf8_decode($data);
                        }
                }
        $this->message[$pos]['cdata'] .= $data;
        // for doclit
        if($this->status == 'header'){
           $this->responseHeaders .= $data;
        } else {
```

```
    $this->document .= $data;
}
    }

    /**
    * get the parsed message (SOAP Body)
    *
    * @return       mixed
    * @access   public
    * @deprecated           use get_soapbody instead
    */
    function get_response(){
            return $this->soapresponse;
    }
    /**
    * get the parsed SOAP Body (NULL if there was none)
    *
    * @return       mixed
    * @access   public
    */
```

```
function get_soapbody(){
        return $this->soapresponse;
}
/**
* get the parsed SOAP Header (NULL if there was none)
*
* @return       mixed
* @access   public
*/
function get_soapheader(){
        return $this->soapheader;
}
/**
* get the unparsed SOAP Header
*
* @return       string XML or empty if no Header
* @access   public
*/
function getHeaders(){
   return $this->responseHeaders;
```

```
        }

        /**
        * decodes simple types into PHP variables
        *
        * @param    string $value value to decode
        * @param    string $type XML type to decode
        * @param    string $typens XML type namespace to decode
        * @return       mixed PHP value
        * @access   private
        */
        function decodeSimple($value, $type, $typens) {
                // TODO: use the namespace!
                if ((!isset($type)) || $type == 'string' || $type == 'long' || $type ==
'unsignedLong') {
                        return (string) $value;
                }
                if ($type == 'int' || $type == 'integer' || $type == 'short' || $type ==
'byte') {
                        return (int) $value;
                }
                if ($type == 'float' || $type == 'double' || $type == 'decimal') {
```

```
            return (double) $value;
    }
    if ($type == 'boolean') {
            if (strtolower($value) == 'false' || strtolower($value) == 'f') {
                    return false;
            }
            return (boolean) $value;
    }
    if ($type == 'base64' || $type == 'base64Binary') {
            $this->debug('Decode base64 value');
            return base64_decode($value);
    }
    // obscure numeric types
    if ($type == 'nonPositiveInteger' || $type == 'negativeInteger'
            || $type == 'nonNegativeInteger' || $type == 'positiveInteger'
            || $type == 'unsignedInt'
            || $type == 'unsignedShort' || $type == 'unsignedByte') {
            return (int) $value;
    }
    // bogus: parser treats array with no elements as a simple type
```

```
        if ($type == 'array') {
                return array();
        }
        // everything else
        return (string) $value;
}

/**
* builds response structures for compound values (arrays/structs)
* and scalars
*
* @param    integer $pos position in node tree
* @return       mixed  PHP value
* @access   private
*/
function buildVal($pos){
        if(!isset($this->message[$pos]['type'])){
                $this->message[$pos]['type'] = '';
        }
```

```
		$this->debug('in buildVal() for '.$this->message[$pos]['name']."(pos $pos) of type ".$this->message[$pos]['type']);
		// if there are children...
		if($this->message[$pos]['children'] != ''){
			$this->debug('in buildVal, there are children');
			$children = explode('|',$this->message[$pos]['children']);
			array_shift($children); // knock off empty
			// md array
			if(isset($this->message[$pos]['arrayCols']) && $this->message[$pos]['arrayCols'] != ''){
		$r=0; // rowcount
		$c=0; // colcount
		foreach($children as $child_pos){
					$this->debug("in buildVal, got an MD array element: $r, $c");
					$params[$r][] = $this->message[$child_pos]['result'];
				$c++;
				if($c == $this->message[$pos]['arrayCols']){
				$c = 0;
					$r++;
```

```
                                    }
            }
        // array
                                } elseif($this->message[$pos]['type'] == 'array' || $this-
>message[$pos]['type'] == 'Array'){
            $this->debug('in buildVal, adding array '.$this->message[$pos]['name']);
            foreach($children as $child_pos){
                $params[] = &$this->message[$child_pos]['result'];
            }
        // apache Map type: java hashtable
        } elseif($this->message[$pos]['type'] == 'Map' && $this-
>message[$pos]['type_namespace'] == 'http://xml.apache.org/xml-soap'){
            $this->debug('in buildVal, Java Map '.$this->message[$pos]['name']);
            foreach($children as $child_pos){
                $kv = explode("|",$this->message[$child_pos]['children']);
                $params[$this->message[$kv[1]]['result']] = &$this-
>message[$kv[2]]['result'];
            }
        // generic compound type
        //} elseif($this->message[$pos]['type'] == 'SOAPStruct' || $this-
>message[$pos]['type'] == 'struct') {
```

```
                } else {
                    // Apache Vector type: treat as an array
            $this->debug('in buildVal, adding Java Vector or generic compound type '.$this->message[$pos]['name']);
                    if ($this->message[$pos]['type'] == 'Vector' && $this->message[$pos]['type_namespace'] == 'http://xml.apache.org/xml-soap') {
                        $notstruct = 1;
                    } else {
                        $notstruct = 0;
                }
                //
                foreach($children as $child_pos){
                    if($notstruct){
                        $params[] = &$this->message[$child_pos]['result'];
                    } else {
                        if (isset($params[$this->message[$child_pos]['name']])) {
                            // de-serialize repeated element name into an array
                            if ((!is_array($params[$this->message[$child_pos]['name']])) || (!isset($params[$this->message[$child_pos]['name']][0]))) {
```

```
                                        $params[$this-
>message[$child_pos]['name']] = array($params[$this-
>message[$child_pos]['name']]);

                                    }

                                    $params[$this->message[$child_pos]['name']][]
= &$this->message[$child_pos]['result'];

                                } else {

                                    $params[$this-
>message[$child_pos]['name']] = &$this->message[$child_pos]['result'];

                                }

                            }

                        }

                        if (isset($this->message[$pos]['xattrs'])) {

                    $this->debug('in buildVal, handling attributes');

                            foreach ($this->message[$pos]['xattrs'] as $n => $v) {

                                $params[$n] = $v;

                            }

                        }

                        // handle simpleContent

                        if (isset($this->message[$pos]['cdata']) && trim($this-
>message[$pos]['cdata']) != '') {
```

```
            $this->debug('in buildVal, handling simpleContent');
                if (isset($this->message[$pos]['type'])) {
                                    $params['!'] = $this->decodeSimple($this->message[$pos]['cdata'], $this->message[$pos]['type'], isset($this->message[$pos]['type_namespace']) ? $this->message[$pos]['type_namespace'] : '');
                                } else {
                                    $parent = $this->message[$pos]['parent'];
                                    if (isset($this->message[$parent]['type']) && ($this->message[$parent]['type'] == 'array') && isset($this->message[$parent]['arrayType'])) {
                                        $params['!'] = $this->decodeSimple($this->message[$pos]['cdata'], $this->message[$parent]['arrayType'], isset($this->message[$parent]['arrayTypeNamespace']) ? $this->message[$parent]['arrayTypeNamespace'] : '');
                                    } else {
                                        $params['!'] = $this->message[$pos]['cdata'];
                                    }
                                }
                            }
                            $ret = is_array($params) ? $params : array();
                            $this->debug('in buildVal, return:');
                            $this->appendDebug($this->varDump($ret));
```

```
                return $ret;
            } else {
        $this->debug('in buildVal, no children, building scalar');
                $cdata = isset($this->message[$pos]['cdata']) ? $this->message[$pos]['cdata'] : '';
        if (isset($this->message[$pos]['type'])) {
                    $ret = $this->decodeSimple($cdata, $this->message[$pos]['type'], isset($this->message[$pos]['type_namespace']) ? $this->message[$pos]['type_namespace'] : '');
                    $this->debug("in buildVal, return: $ret");
                    return $ret;
                }
                $parent = $this->message[$pos]['parent'];
                if (isset($this->message[$parent]['type']) && ($this->message[$parent]['type'] == 'array') && isset($this->message[$parent]['arrayType'])) {
                    $ret = $this->decodeSimple($cdata, $this->message[$parent]['arrayType'], isset($this->message[$parent]['arrayTypeNamespace']) ? $this->message[$parent]['arrayTypeNamespace'] : '');
                    $this->debug("in buildVal, return: $ret");
                    return $ret;
                }
```

```
        $ret = $this->message[$pos]['cdata'];
                        $this->debug("in buildVal, return: $ret");
        return $ret;
                }
        }
}


/**
 * Backward compatibility
 */
class soap_parser extends nusoap_parser {
}


?><?php




/**
*
* [nu]soapclient higher level class for easy usage.
```

```
*
* usage:
*
* // instantiate client with server info
* $soapclient = new nusoap_client( string path [ ,mixed wsdl] );
*
* // call method, get results
* echo $soapclient->call( string methodname [ ,array parameters] );
*
* // bye bye client
* unset($soapclient);
*
* @author   Dietrich Ayala <dietrich@ganx4.com>
* @author   Scott Nichol <snichol@users.sourceforge.net>
* @version  $Id: nusoap.php,v 1.123 2010/04/26 20:15:08 snichol Exp $
* @access   public
*/
class nusoap_client extends nusoap_base  {
```

```
        var $username = '';                     // Username for HTTP
authentication

        var $password = '';                     // Password for HTTP
authentication

        var $authtype = '';                     // Type of HTTP authentication

        var $certRequest = array();             // Certificate for HTTP SSL
authentication

        var $requestHeaders = false;  // SOAP headers in request (text)

        var $responseHeaders = '';              // SOAP headers from response
(incomplete namespace resolution) (text)

        var $responseHeader = NULL;             // SOAP Header from response
(parsed)

        var $document = '';                     // SOAP body response portion
(incomplete namespace resolution) (text)

        var $endpoint;

        var $forceEndpoint = '';                // overrides WSDL endpoint

    var $proxyhost = '';

    var $proxyport = '';

        var $proxyusername = '';

        var $proxypassword = '';

        var $portName = '';                     // port name to use in WSDL
```

```
   var $xml_encoding = '';                    // character set encoding of incoming
(response) messages
        var $http_encoding = false;
        var $timeout = 0;                          // HTTP connection timeout
        var $response_timeout = 30;          // HTTP response timeout
        var $endpointType = '';                    // soap|wsdl, empty for WSDL
initialization error
        var $persistentConnection = false;
        var $defaultRpcParams = false;       // This is no longer used
        var $request = '';                         // HTTP request
        var $response = '';                        // HTTP response
        var $responseData = '';                    // SOAP payload of response
        var $cookies = array();                    // Cookies from response or for
request
   var $decode_utf8 = true;             // toggles whether the parser decodes element
content w/ utf8_decode()
        var $operations = array();           // WSDL operations, empty for WSDL
initialization error
        var $curl_options = array();   // User-specified cURL options
        var $bindingType = '';                     // WSDL operation binding type
        var $use_curl = false;                     // whether to always try to use cURL
```

```
/*
 * fault related variables
 */
/**
 * @var      fault
 * @access   public
 */
var $fault;
/**
 * @var      faultcode
 * @access   public
 */
var $faultcode;
/**
 * @var      faultstring
 * @access   public
 */
var $faultstring;
/**
 * @var      faultdetail
```

```
     * @access   public
     */
    var $faultdetail;

    /**
    * constructor
    *
    * @param    mixed $endpoint SOAP server or WSDL URL (string), or wsdl instance (object)
    * @param    mixed $wsdl optional, set to 'wsdl' or true if using WSDL
    * @param    string $proxyhost optional
    * @param    string $proxyport optional
    * @param      string $proxyusername optional
    * @param      string $proxypassword optional
    * @param      integer $timeout set the connection timeout
    * @param      integer $response_timeout set the response timeout
    * @param      string $portName optional portName in WSDL document
    * @access   public
    */
```

```
        function nusoap_client($endpoint,$wsdl = false,$proxyhost = false,$proxyport
= false,$proxyusername = false, $proxypassword = false, $timeout = 0,
$response_timeout = 30, $portName = ''){
                parent::nusoap_base();
                $this->endpoint = $endpoint;
                $this->proxyhost = $proxyhost;
                $this->proxyport = $proxyport;
                $this->proxyusername = $proxyusername;
                $this->proxypassword = $proxypassword;
                $this->timeout = $timeout;
                $this->response_timeout = $response_timeout;
                $this->portName = $portName;

                $this->debug("ctor wsdl=$wsdl timeout=$timeout
response_timeout=$response_timeout");
                $this->appendDebug('endpoint=' . $this->varDump($endpoint));

                // make values
                if($wsdl){
                        if (is_object($endpoint) && (get_class($endpoint) == 'wsdl')) {
                                $this->wsdl = $endpoint;
```

```
                $this->endpoint = $this->wsdl->wsdl;
                $this->wsdlFile = $this->endpoint;
                $this->debug('existing wsdl instance created from ' .
$this->endpoint);
                $this->checkWSDL();
            } else {
                $this->wsdlFile = $this->endpoint;
                $this->wsdl = null;
                $this->debug('will use lazy evaluation of wsdl from ' .
$this->endpoint);
            }
            $this->endpointType = 'wsdl';
        } else {
            $this->debug("instantiate SOAP with endpoint at $endpoint");
            $this->endpointType = 'soap';
        }
    }
    /**
    * calls method, returns PHP native type
    *
    * @param    string $operation SOAP server URL or path
```

```
     * @param    mixed $params An array, associative or simple, of the parameters
     *                         for the method call, or a string that is the XML
     *                         for the call.  For rpc style, this call will
     *                         wrap the XML in a tag named after the method, as
     *                         well as the SOAP Envelope and Body.  For document
     *                         style, this will only wrap with the Envelope and Body.
     *                         IMPORTANT: when using an array with document style,
     *                         in which case there
     *                    is really one parameter, the root of the fragment
     *                    used in the call, which encloses what programmers
     *                    normally think of parameters.  A parameter array
     *                    *must* include the wrapper.
     * @param    string $namespace optional method namespace (WSDL can override)
     * @param    string $soapAction optional SOAPAction value (WSDL can override)
     * @param    mixed $headers optional string of XML with SOAP header content, or array of soapval objects for SOAP headers, or associative array
```

```
    * @param       boolean $rpcParams optional (no longer used)
    * @param       string   $style optional (rpc|document) the style to use when serializing parameters (WSDL can override)
    * @param       string   $use optional (encoded|literal) the use when serializing parameters (WSDL can override)
    * @return      mixed  response from SOAP call, normally an associative array mirroring the structure of the XML response, false for certain fatal errors
    * @access   public
    */
    function call($operation,$params=array(),$namespace='http://tempuri.org',$soapAction='',$headers=false,$rpcParams=null,$style='rpc',$use='encoded'){
        $this->operation = $operation;
        $this->fault = false;
        $this->setError('');
        $this->request = '';
        $this->response = '';
        $this->responseData = '';
        $this->faultstring = '';
        $this->faultcode = '';
        $this->opData = array();
```

```
        $this->debug("call: operation=$operation, namespace=$namespace,
soapAction=$soapAction, rpcParams=$rpcParams, style=$style, use=$use,
endpointType=$this->endpointType");
        $this->appendDebug('params=' . $this->varDump($params));
        $this->appendDebug('headers=' . $this->varDump($headers));
        if ($headers) {
            $this->requestHeaders = $headers;
        }
        if ($this->endpointType == 'wsdl' && is_null($this->wsdl)) {
            $this->loadWSDL();
            if ($this->getError())
                return false;
        }
        // serialize parameters
        if($this->endpointType == 'wsdl' && $opData = $this-
>getOperationData($operation)){
            // use WSDL for operation
            $this->opData = $opData;
            $this->debug("found operation");
            $this->appendDebug('opData=' . $this->varDump($opData));
            if (isset($opData['soapAction'])) {
```

```
                $soapAction = $opData['soapAction'];
            }
            if (! $this->forceEndpoint) {
                $this->endpoint = $opData['endpoint'];
            } else {
                $this->endpoint = $this->forceEndpoint;
            }
            $namespace = isset($opData['input']['namespace']) ?
$opData['input']['namespace'] :      $namespace;
            $style = $opData['style'];
            $use = $opData['input']['use'];
            // add ns to ns array
            if($namespace != '' && !isset($this->wsdl-
>namespaces[$namespace])){
                $nsPrefix = 'ns' . rand(1000, 9999);
                $this->wsdl->namespaces[$nsPrefix] = $namespace;
            }
        $nsPrefix = $this->wsdl->getPrefixFromNamespace($namespace);
            // serialize payload
            if (is_string($params)) {
```

```
                $this->debug("serializing param string for WSDL
operation $operation");
                $payload = $params;
            } elseif (is_array($params)) {
                $this->debug("serializing param array for WSDL
operation $operation");
                $payload = $this->wsdl-
>serializeRPCParameters($operation,'input',$params,$this->bindingType);
            } else {
                $this->debug('params must be array or string');
                $this->setError('params must be array or string');
                return false;
            }
        $usedNamespaces = $this->wsdl->usedNamespaces;
            if (isset($opData['input']['encodingStyle'])) {
                $encodingStyle = $opData['input']['encodingStyle'];
            } else {
                $encodingStyle = '';
            }
            $this->appendDebug($this->wsdl->getDebug());
            $this->wsdl->clearDebug();
```

```
            if ($errstr = $this->wsdl->getError()) {
                $this->debug('got wsdl error: '.$errstr);
                $this->setError('wsdl error: '.$errstr);
                return false;
            }
        } elseif($this->endpointType == 'wsdl') {
            // operation not in WSDL
            $this->appendDebug($this->wsdl->getDebug());
            $this->wsdl->clearDebug();
            $this->setError('operation '.$operation.' not present in WSDL.');
            $this->debug("operation '$operation' not present in WSDL.");
            return false;
        } else {
            // no WSDL
            //$this->namespaces['ns1'] = $namespace;
            $nsPrefix = 'ns' . rand(1000, 9999);
            // serialize
            $payload = '';
            if (is_string($params)) {
```

```
                    $this->debug("serializing param string for operation
$operation");
                    $payload = $params;
                } elseif (is_array($params)) {
                    $this->debug("serializing param array for operation
$operation");
                    foreach($params as $k => $v){
                        $payload .= $this-
>serialize_val($v,$k,false,false,false,false,$use);
                    }
                } else {
                    $this->debug('params must be array or string');
                    $this->setError('params must be array or string');
                    return false;
                }
                $usedNamespaces = array();
                if ($use == 'encoded') {
                    $encodingStyle =
'http://schemas.xmlsoap.org/soap/encoding/';
                } else {
                    $encodingStyle = '';
```

```
                }
            }
            // wrap RPC calls with method element
            if ($style == 'rpc') {
                if ($use == 'literal') {
                    $this->debug("wrapping RPC request with literal
method element");
                    if ($namespace) {
                        // http://www.ws-i.org/Profiles/BasicProfile-1.1-
2004-08-24.html R2735 says rpc/literal accessor elements should not be in a
namespace
                        $payload = "<$nsPrefix:$operation
xmlns:$nsPrefix=\"$namespace\">" .
                                $payload .
                                "</$nsPrefix:$operation>";
                    } else {
                        $payload = "<$operation>" . $payload .
"</$operation>";
                    }
                } else {
                    $this->debug("wrapping RPC request with encoded
method element");
```

```
                    if ($namespace) {
                        $payload = "<$nsPrefix:$operation
xmlns:$nsPrefix=\"$namespace\">" .
                                $payload .
                                "</$nsPrefix:$operation>";
                    } else {
                        $payload = "<$operation>" .
                                $payload .
                                "</$operation>";
                    }
                }
            }
            // serialize envelope
            $soapmsg = $this->serializeEnvelope($payload,$this-
>requestHeaders,$usedNamespaces,$style,$use,$encodingStyle);
            $this->debug("endpoint=$this->endpoint, soapAction=$soapAction,
namespace=$namespace, style=$style, use=$use, encodingStyle=$encodingStyle");
            $this->debug('SOAP message length=' . strlen($soapmsg) . ' contents
(max 1000 bytes)=' . substr($soapmsg, 0, 1000));
            // send
            $return = $this->send($this-
>getHTTPBody($soapmsg),$soapAction,$this->timeout,$this->response_timeout);
```

```
                if($errstr = $this->getError()){
                        $this->debug('Error: '.$errstr);
                        return false;
                } else {
                        $this->return = $return;
                        $this->debug('sent message successfully and got a(n)
'.gettype($return));
        $this->appendDebug('return=' . $this->varDump($return));
                        // fault?
                        if(is_array($return) && isset($return['faultcode'])){
                                $this->debug('got fault');
                                $this->setError($return['faultcode'].':
'.$return['faultstring']);
                                $this->fault = true;
                                foreach($return as $k => $v){
                                        $this->$k = $v;
                                        $this->debug("$k = $v<br>");
                                }
                                return $return;
                        } elseif ($style == 'document') {
```

```
                    // NOTE: if the response is defined to have multiple
parts (i.e. unwrapped),
                    // we are only going to return the first part here...sorry
about that
                    return $return;
                } else {
                    // array of return values
                    if(is_array($return)){
                        // multiple 'out' parameters, which we return
wrapped up
                        // in the array
                        if(sizeof($return) > 1){
                            return $return;
                        }
                        // single 'out' parameter (normally the return
value)
                        $return = array_shift($return);
                        $this->debug('return shifted value: ');
                        $this->appendDebug($this->varDump($return));
                return $return;
                    // nothing returned (ie, echoVoid)
```

```
                        } else {
                                return "";
                        }
                }
        }
}

/**
* check WSDL passed as an instance or pulled from an endpoint
*
* @access   private
*/
function checkWSDL() {
        $this->appendDebug($this->wsdl->getDebug());
        $this->wsdl->clearDebug();
        $this->debug('checkWSDL');
        // catch errors
        if ($errstr = $this->wsdl->getError()) {
                $this->appendDebug($this->wsdl->getDebug());
                $this->wsdl->clearDebug();
```

```
                $this->debug('got wsdl error: '.$errstr);
                $this->setError('wsdl error: '.$errstr);
            } elseif ($this->operations = $this->wsdl->getOperations($this->portName, 'soap')) {
                $this->appendDebug($this->wsdl->getDebug());
                $this->wsdl->clearDebug();
                $this->bindingType = 'soap';
                $this->debug('got '.count($this->operations).' operations from wsdl '.$this->wsdlFile.' for binding type '.$this->bindingType);
            } elseif ($this->operations = $this->wsdl->getOperations($this->portName, 'soap12')) {
                $this->appendDebug($this->wsdl->getDebug());
                $this->wsdl->clearDebug();
                $this->bindingType = 'soap12';
                $this->debug('got '.count($this->operations).' operations from wsdl '.$this->wsdlFile.' for binding type '.$this->bindingType);
                $this->debug('****************** WARNING: SOAP 1.2 BINDING ******************');
            } else {
                $this->appendDebug($this->wsdl->getDebug());
                $this->wsdl->clearDebug();
                $this->debug('getOperations returned false');
```

```
                    $this->setError('no operations defined in the WSDL
document!');
                }
        }

        /**
         * instantiate wsdl object and parse wsdl file
         *
         * @access      public
         */
        function loadWSDL() {
                $this->debug('instantiating wsdl class with doc: '.$this->wsdlFile);
                $this->wsdl = new wsdl('',$this->proxyhost,$this->proxyport,$this-
>proxyusername,$this->proxypassword,$this->timeout,$this-
>response_timeout,$this->curl_options,$this->use_curl);
                $this->wsdl->setCredentials($this->username, $this->password, $this-
>authtype, $this->certRequest);
                $this->wsdl->fetchWSDL($this->wsdlFile);
                $this->checkWSDL();
        }
```

```
    /**
    * get available data pertaining to an operation
    *
    * @param    string $operation operation name
    * @return       array array of data pertaining to the operation
    * @access   public
    */
    function getOperationData($operation){
        if ($this->endpointType == 'wsdl' && is_null($this->wsdl)) {
            $this->loadWSDL();
            if ($this->getError())
                return false;
        }
        if(isset($this->operations[$operation])){
            return $this->operations[$operation];
        }
        $this->debug("No data for operation: $operation");
    }
/**
* send the SOAP message
```

```
*
* Note: if the operation has multiple return values
* the return value of this method will be an array
* of those values.
*
    * @param    string $msg a SOAPx4 soapmsg object
    * @param    string $soapaction SOAPAction value
    * @param    integer $timeout set connection timeout in seconds
    * @param      integer $response_timeout set response timeout in seconds
    * @return       mixed native PHP types.
    * @access   private
    */
    function send($msg, $soapaction = '', $timeout=0, $response_timeout=30) {
            $this->checkCookies();
            // detect transport
            switch(true){
                    // http(s)
                    case preg_match('/^http/',$this->endpoint):
                            $this->debug('transporting via HTTP');
```

```
                    if($this->persistentConnection == true &&
is_object($this->persistentConnection)){
                        $http =& $this->persistentConnection;
                    } else {
                        $http = new soap_transport_http($this-
>endpoint, $this->curl_options, $this->use_curl);
                        if ($this->persistentConnection) {
                            $http->usePersistentConnection();
                        }
                    }
                    $http->setContentType($this->getHTTPContentType(),
$this->getHTTPContentTypeCharset());
                    $http->setSOAPAction($soapaction);
                    if($this->proxyhost && $this->proxyport){
                        $http->setProxy($this->proxyhost,$this-
>proxyport,$this->proxyusername,$this->proxypassword);
                    }
                if($this->authtype != '') {
                        $http->setCredentials($this->username, $this-
>password, $this->authtype, array(), $this->certRequest);
                    }
                    if($this->http_encoding != ''){
```

```
                $http->setEncoding($this->http_encoding);
            }
            $this->debug('sending message, length='.strlen($msg));
            if(preg_match('/^http:/',$this->endpoint)){
            //if(strpos($this->endpoint,'http:')){
                $this->responseData = $http->send($msg,$timeout,$response_timeout,$this->cookies);
            } elseif(preg_match('/^https/',$this->endpoint)){
            //} elseif(strpos($this->endpoint,'https:')){
                //if(phpversion() == '4.3.0-dev'){
                    //$response = $http->send($msg,$timeout,$response_timeout);
                //$this->request = $http->outgoing_payload;
                    //$this->response = $http->incoming_payload;
                //} else
                $this->responseData = $http->sendHTTPS($msg,$timeout,$response_timeout,$this->cookies);
            } else {
                $this->setError('no http/s in endpoint url');
            }
```

```
$this->request = $http->outgoing_payload;
$this->response = $http->incoming_payload;
$this->appendDebug($http->getDebug());
$this->UpdateCookies($http->incoming_cookies);

// save transport object if using persistent connections
if ($this->persistentConnection) {
        $http->clearDebug();
        if (!is_object($this->persistentConnection)) {
                $this->persistentConnection = $http;
        }
}

if($err = $http->getError()){
        $this->setError('HTTP Error: '.$err);
        return false;
} elseif($this->getError()){
        return false;
} else {
```

```
                                    $this->debug('got response, length='.
strlen($this->responseData).' type='.$http->incoming_headers['content-type']);
                                    return $this->parseResponse($http-
>incoming_headers, $this->responseData);
                                }
                            break;
                            default:
                                $this->setError('no transport found, or selected transport
is not yet supported!');
                            return false;
                            break;
                    }
            }

            /**
            * processes SOAP message returned from server
            *
            * @param      array   $headers        The HTTP headers
            * @param      string  $data           unprocessed response data from server
            * @return     mixed   value of the message, decoded into a PHP type
            * @access   private
```

```
        */
    function parseResponse($headers, $data) {
                $this->debug('Entering parseResponse() for data of length ' .
strlen($data) . ' headers:');
                $this->appendDebug($this->varDump($headers));
        if (!isset($headers['content-type'])) {
                        $this->setError('Response not of type text/xml (no content-type
header)');
                        return false;
        }
                if (!strstr($headers['content-type'], 'text/xml')) {
                        $this->setError('Response not of type text/xml: ' .
$headers['content-type']);
                        return false;
                }
                if (strpos($headers['content-type'], '=')) {
                        $enc = str_replace("", '', substr(strstr($headers["content-type"],
'='), 1));
                        $this->debug('Got response encoding: ' . $enc);
                        if(preg_match('/^(ISO-8859-1|US-ASCII|UTF-8)$/i',$enc)){
                                $this->xml_encoding = strtoupper($enc);
```

```
                } else {
                        $this->xml_encoding = 'US-ASCII';
                }
        } else {
                // should be US-ASCII for HTTP 1.0 or ISO-8859-1 for HTTP 1.1
                $this->xml_encoding = 'ISO-8859-1';
        }
        $this->debug('Use encoding: ' . $this->xml_encoding . ' when creating nusoap_parser');
        $parser = new nusoap_parser($data,$this->xml_encoding,$this->operation,$this->decode_utf8);
        // add parser debug data to our debug
        $this->appendDebug($parser->getDebug());
        // if parse errors
        if($errstr = $parser->getError()){
                $this->setError( $errstr);
                // destroy the parser object
                unset($parser);
                return false;
        } else {
```

```
                    // get SOAP headers
                    $this->responseHeaders = $parser->getHeaders();
                    // get SOAP headers
                    $this->responseHeader = $parser->get_soapheader();
                    // get decoded message
                    $return = $parser->get_soapbody();
// add document for doclit support
$this->document = $parser->document;
                    // destroy the parser object
                    unset($parser);
                    // return decode message
                    return $return;
            }
 }


/**
* sets user-specified cURL options
*
* @param      mixed $option The cURL option (always integer?)
* @param      mixed $value The cURL option value
```

```
        * @access   public
        */
        function setCurlOption($option, $value) {
                $this->debug("setCurlOption option=$option, value=");
                $this->appendDebug($this->varDump($value));
                $this->curl_options[$option] = $value;
        }


        /**
        * sets the SOAP endpoint, which can override WSDL
        *
        * @param       string $endpoint The endpoint URL to use, or empty string or false to prevent override
        * @access   public
        */
        function setEndpoint($endpoint) {
                $this->debug("setEndpoint(\"$endpoint\")");
                $this->forceEndpoint = $endpoint;
        }
```

```
    /**
    * set the SOAP headers
    *
    * @param      mixed $headers String of XML with SOAP header content, or array of soapval objects for SOAP headers
    * @access   public
    */
    function setHeaders($headers){
        $this->debug("setHeaders headers=");
        $this->appendDebug($this->varDump($headers));
        $this->requestHeaders = $headers;
    }


    /**
    * get the SOAP response headers (namespace resolution incomplete)
    *
    * @return       string
    * @access   public
    */
    function getHeaders(){
```

```
        return $this->responseHeaders;
}

/**
* get the SOAP response Header (parsed)
*
* @return       mixed
* @access   public
*/
function getHeader(){
        return $this->responseHeader;
}

/**
* set proxy info here
*
* @param    string $proxyhost
* @param    string $proxyport
* @param       string $proxyusername
* @param       string $proxypassword
```

```
     * @access   public
     */
     function setHTTPProxy($proxyhost, $proxyport, $proxyusername = '', $proxypassword = '') {
          $this->proxyhost = $proxyhost;
          $this->proxyport = $proxyport;
          $this->proxyusername = $proxyusername;
          $this->proxypassword = $proxypassword;
     }

     /**
     * if authenticating, set user credentials here
     *
     * @param    string $username
     * @param    string $password
     * @param       string $authtype (basic|digest|certificate|ntlm)
     * @param       array $certRequest (keys must be cainfofile (optional), sslcertfile, sslkeyfile, passphrase, verifypeer (optional), verifyhost (optional): see corresponding options in cURL docs)
     * @access   public
     */
```

```
    function setCredentials($username, $password, $authtype = 'basic', $certRequest = array()) {
        $this->debug("setCredentials username=$username authtype=$authtype certRequest=");
        $this->appendDebug($this->varDump($certRequest));
        $this->username = $username;
        $this->password = $password;
        $this->authtype = $authtype;
        $this->certRequest = $certRequest;
    }

    /**
    * use HTTP encoding
    *
    * @param    string $enc HTTP encoding
    * @access   public
    */
    function setHTTPEncoding($enc='gzip, deflate'){
        $this->debug("setHTTPEncoding(\"$enc\")");
        $this->http_encoding = $enc;
    }
```

```
/**
* Set whether to try to use cURL connections if possible
*
* @param       boolean $use Whether to try to use cURL
* @access   public
*/
function setUseCURL($use) {
        $this->debug("setUseCURL($use)");
        $this->use_curl = $use;
}

/**
* use HTTP persistent connections if possible
*
* @access   public
*/
function useHTTPPersistentConnection(){
        $this->debug("useHTTPPersistentConnection");
        $this->persistentConnection = true;
```

```
}

/**
* gets the default RPC parameter setting.
* If true, default is that call params are like RPC even for document style.
* Each call() can override this value.
*
* This is no longer used.
*
* @return boolean
* @access public
* @deprecated
*/
function getDefaultRpcParams() {
        return $this->defaultRpcParams;
}

/**
* sets the default RPC parameter setting.
* If true, default is that call params are like RPC even for document style
```

```
* Each call() can override this value.
*
* This is no longer used.
*
* @param    boolean $rpcParams
* @access public
* @deprecated
*/
function setDefaultRpcParams($rpcParams) {
        $this->defaultRpcParams = $rpcParams;
}

/**
* dynamically creates an instance of a proxy class,
* allowing user to directly call methods from wsdl
*
* @return   object soap_proxy object
* @access   public
*/
function getProxy() {
```

```
                $r = rand();
                $evalStr = $this->_getProxyClassCode($r);
                //$this->debug("proxy class: $evalStr");
                if ($this->getError()) {
                        $this->debug("Error from _getProxyClassCode, so return
NULL");
                        return null;
                }
                // eval the class
                eval($evalStr);
                // instantiate proxy object
                eval("\$proxy = new nusoap_proxy_$r();");
                // transfer current wsdl data to the proxy thereby avoiding parsing the
wsdl twice
                $proxy->endpointType = 'wsdl';
                $proxy->wsdlFile = $this->wsdlFile;
                $proxy->wsdl = $this->wsdl;
                $proxy->operations = $this->operations;
                $proxy->defaultRpcParams = $this->defaultRpcParams;
                // transfer other state
                $proxy->soap_defencoding = $this->soap_defencoding;
```

```
$proxy->username = $this->username;
$proxy->password = $this->password;
$proxy->authtype = $this->authtype;
$proxy->certRequest = $this->certRequest;
$proxy->requestHeaders = $this->requestHeaders;
$proxy->endpoint = $this->endpoint;
$proxy->forceEndpoint = $this->forceEndpoint;
$proxy->proxyhost = $this->proxyhost;
$proxy->proxyport = $this->proxyport;
$proxy->proxyusername = $this->proxyusername;
$proxy->proxypassword = $this->proxypassword;
$proxy->http_encoding = $this->http_encoding;
$proxy->timeout = $this->timeout;
$proxy->response_timeout = $this->response_timeout;
$proxy->persistentConnection = &$this->persistentConnection;
$proxy->decode_utf8 = $this->decode_utf8;
$proxy->curl_options = $this->curl_options;
$proxy->bindingType = $this->bindingType;
$proxy->use_curl = $this->use_curl;
return $proxy;
```

```
}

/**
* dynamically creates proxy class code
*
* @return   string PHP/NuSOAP code for the proxy class
* @access   private
*/
function _getProxyClassCode($r) {
        $this->debug("in getProxy endpointType=$this->endpointType");
        $this->appendDebug("wsdl=" . $this->varDump($this->wsdl));
        if ($this->endpointType != 'wsdl') {
                $evalStr = 'A proxy can only be created for a WSDL client';
                $this->setError($evalStr);
                $evalStr = "echo \"$evalStr\";";
                return $evalStr;
        }
        if ($this->endpointType == 'wsdl' && is_null($this->wsdl)) {
                $this->loadWSDL();
                if ($this->getError()) {
```

```
                return "echo \"" . $this->getError() . "\";";
            }
        }
        $evalStr = '';
        foreach ($this->operations as $operation => $opData) {
            if ($operation != '') {
                // create param string and param comment string
                if (sizeof($opData['input']['parts']) > 0) {
                    $paramStr = '';
                    $paramArrayStr = '';
                    $paramCommentStr = '';
                    foreach ($opData['input']['parts'] as $name =>
$type) {
                        $paramStr .= "\$$name, ";
                        $paramArrayStr .= "'$name' => \$$name,
";
                        $paramCommentStr .= "$type \$$name,
";
                    }
                    $paramStr = substr($paramStr, 0,
strlen($paramStr)-2);
```

```
                                        $paramArrayStr = substr($paramArrayStr, 0,
strlen($paramArrayStr)-2);
                                        $paramCommentStr =
substr($paramCommentStr, 0, strlen($paramCommentStr)-2);
                                    } else {
                                        $paramStr = '';
                                        $paramArrayStr = '';
                                        $paramCommentStr = 'void';
                                    }
                                    $opData['namespace'] = !isset($opData['namespace']) ?
'http://testuri.com' : $opData['namespace'];
                                    $evalStr .= "// $paramCommentStr
        function " . str_replace('.', '__', $operation) . "($paramStr) {
                \$params = array($paramArrayStr);
                return \$this->call('$operation', \$params, '".$opData['namespace']."',
'".(isset($opData['soapAction']) ? $opData['soapAction'] : '')."');
        }
        ";
                                    unset($paramStr);
                                    unset($paramCommentStr);
                                }
```

```
                }

                $evalStr = 'class nusoap_proxy_'.$r.' extends nusoap_client {

        '.$evalStr.'

};';

                return $evalStr;

        }


        /**

        * dynamically creates proxy class code

        *

        * @return   string PHP/NuSOAP code for the proxy class

        * @access   public

        */

        function getProxyClassCode() {

                $r = rand();

                return $this->_getProxyClassCode($r);

        }


        /**

        * gets the HTTP body for the current request.
```

```
*
* @param string $soapmsg The SOAP payload
* @return string The HTTP body, which includes the SOAP payload
* @access private
*/
function getHTTPBody($soapmsg) {
        return $soapmsg;
}

/**
* gets the HTTP content type for the current request.
*
* Note: getHTTPBody must be called before this.
*
* @return string the HTTP content type for the current request.
* @access private
*/
function getHTTPContentType() {
        return 'text/xml';
}
```

```
        /**
        * gets the HTTP content type charset for the current request.
        * returns false for non-text content types.
        *
        * Note: getHTTPBody must be called before this.
        *
        * @return string the HTTP content type charset for the current request.
        * @access private
        */
        function getHTTPContentTypeCharset() {
                return $this->soap_defencoding;
        }


        /*
        * whether or not parser should decode utf8 element content
*
* @return   always returns true
* @access   public
*/
```

```
function decodeUTF8($bool){
		$this->decode_utf8 = $bool;
		return true;
}


	/**
	 * adds a new Cookie into $this->cookies array
	 *
	 * @param	string $name Cookie Name
	 * @param	string $value Cookie Value
	 * @return	boolean if cookie-set was successful returns true, else false
	 * @access	public
	 */
	function setCookie($name, $value) {
		if (strlen($name) == 0) {
			return false;
		}
		$this->cookies[] = array('name' => $name, 'value' => $value);
		return true;
	}
```

```
/**
 * gets all Cookies
 *
 * @return   array with all internal cookies
 * @access   public
 */
function getCookies() {
        return $this->cookies;
}

/**
 * checks all Cookies and delete those which are expired
 *
 * @return   boolean always return true
 * @access   private
 */
function checkCookies() {
        if (sizeof($this->cookies) == 0) {
                return true;
```

```
                }
                $this->debug('checkCookie: check ' . sizeof($this->cookies) . '
cookies');
                $curr_cookies = $this->cookies;
                $this->cookies = array();
                foreach ($curr_cookies as $cookie) {
                        if (! is_array($cookie)) {
                                $this->debug('Remove cookie that is not an array');
                                continue;
                        }
                        if ((isset($cookie['expires'])) && (! empty($cookie['expires'])))
{
                                if (strtotime($cookie['expires']) > time()) {
                                        $this->cookies[] = $cookie;
                                } else {
                                        $this->debug('Remove expired cookie ' .
$cookie['name']);
                                }
                        } else {
                                $this->cookies[] = $cookie;
                        }
```

```
                }
                $this->debug('checkCookie: '.sizeof($this->cookies).' cookies left in
array');
                return true;
        }

        /**
         * updates the current cookies with a new set
         *
         * @param      array $cookies new cookies with which to update current ones
         * @return     boolean always return true
         * @access     private
         */
        function UpdateCookies($cookies) {
                if (sizeof($this->cookies) == 0) {
                        // no existing cookies: take whatever is new
                        if (sizeof($cookies) > 0) {
                                $this->debug('Setting new cookie(s)');
                                $this->cookies = $cookies;
                        }
```

```
                return true;
        }
        if (sizeof($cookies) == 0) {
                // no new cookies: keep what we've got
                return true;
        }
        // merge
        foreach ($cookies as $newCookie) {
                if (!is_array($newCookie)) {
                        continue;
                }
                if ((!isset($newCookie['name'])) ||
(!isset($newCookie['value']))) {
                        continue;
                }
                $newName = $newCookie['name'];

                $found = false;
                for ($i = 0; $i < count($this->cookies); $i++) {
                        $cookie = $this->cookies[$i];
```

```
                    if (!is_array($cookie)) {
                        continue;
                    }
                    if (!isset($cookie['name'])) {
                        continue;
                    }
                    if ($newName != $cookie['name']) {
                        continue;
                    }
                    $newDomain = isset($newCookie['domain']) ?
$newCookie['domain'] : 'NODOMAIN';
                    $domain = isset($cookie['domain']) ? $cookie['domain']
: 'NODOMAIN';
                    if ($newDomain != $domain) {
                        continue;
                    }
                    $newPath = isset($newCookie['path']) ?
$newCookie['path'] : 'NOPATH';
                    $path = isset($cookie['path']) ? $cookie['path'] :
'NOPATH';
                    if ($newPath != $path) {
```

```
                                continue;
                            }
                            $this->cookies[$i] = $newCookie;
                            $found = true;
                            $this->debug('Update cookie ' . $newName . '=' .
$newCookie['value']);
                            break;
                        }
                        if (! $found) {
                            $this->debug('Add cookie ' . $newName . '=' .
$newCookie['value']);
                            $this->cookies[] = $newCookie;
                        }
                    }
                    return true;
                }
}

if (!extension_loaded('soap')) {
    /**
```

```
        *       For backwards compatiblity, define soapclient unless the PHP SOAP extension is loaded.
        */
        class soapclient extends nusoap_client {
        }
}
?>


<?php
  /*
        incluye el archivo gapi.class.php que agrega el api para administracion de trafico
        */
  require_once 'gapi.class.php';
        // usuario para  se conecta
  define('ga_web_email', 'fernancarpio@gmail.com');
        // la clave de la cuenta
  define('ga_web_pass', 'nando2');
  // aqui el consultamos al servidor de google el trafico de nuestro  que presente las visitas que tenga
```

```
        $data =
get_analytics_data('59231567',array('date'),array('newVisits','visits'),'','2012-04-
29',date('Y-m-d'));

        $reg=count($data['data']);

        $r=$data['data'];

        //consultamos al servidor de trafico el tipo de explorador y version

        $data1 =
get_analytics_data('59231567',array('browser','browserVersion'),array('pageviews','vi
sits'),'','2012-04-29',date('Y-m-d'));

        //consultamos al servidor de trafico  por la paginas mas visitas

        $data2 =
get_analytics_data('59231567',array('pagePath'),array('visits'),'','2012-04-29',date('Y-
m-d'));

        //print_r($data2);



    function get_analytics_data(

    $profile_id,

    $dimensions=array('browser'),

    $metrics=array('pageviews','visits','UniquePageviews'),

    $sort=null,

    $start_date=null,
```

```
$end_date=null
){
$cache_id = md5(serialize(func_get_args()));
$return = false;
// check if the cache item exists.
$temp_folder = '/tmp/ga/';

if(!is_dir($temp_folder))mkdir($temp_folder);
$filename = $temp_folder.$cache_id;
if(is_file($filename)){ // if cache entry exists
if(filemtime($filename) > (time() - 172800)){ // check if it's older than 2 days
$return = unserialize(file_get_contents($filename)); // grab the cached content.
}
}
                        $ga = new gapi(ga_web_email,ga_web_pass);
                        if(!$sort)$sort = current($dimensions);
                        ini_set('display_errors',true);
                        $ga-
>requestReportData($profile_id,$dimensions,$metrics,$sort,null,$start_date,$end_dat
e,null,100);
                        $return = array();
```

```
                $return['data'] = array();
                foreach($ga->getResults() as $result){
                $data = array();
                foreach($dimensions as $d){
                $data[$d] = (string)$result;
                foreach($metrics as $m){
                $data[$m] = $result->{'get'.$m}();
                }
                }
                $return['data'][] = $data;
                }
                $return['total'] = $ga->getTotalResults();
                foreach($metrics as $m){
                $return[$m] = $ga->{'get'.$m}();
                }
                $return['timestamp'] = $ga->getUpdated();

file_put_contents($filename,serialize($return));
return $return;
}
```

```
  ?>
        <table width="200" border="1" cellpadding="1" cellspacing="0"
bordercolor="#000000" align="center">
    <tr>
      <td>
        <table width="478" border="0" bgcolor="#E9E9E9">
    <tr>
      <td width="472"><strong>VISITAS  </strong></td>
    </tr>
  </table>
        <table width="478" border="0" >
 <tr>
  <td width="161" bgcolor="#CCCCCC"><strong>Fecha </strong></td>
  <td width="115" bgcolor="#CCCCCC"><strong>Nueva Visitas </strong></td>
  <td width="188" bgcolor="#CCCCCC"><strong>Visitas</strong></td>
 </tr>
 <?php
 // presenta los trafico segunda las visitas
  $reg=count($data['data']);
```

```
        $r=$data['data'];
        for($i=0;$i<$reg;$i++){
          $fila=$r[$i];


 ?>
 <tr>
   <td><?php echo $fila['date']; ?></td>
   <td><?php echo $fila['newVisits']; ?></td>
   <td><?php echo $fila['visits']; ?></td>
 </tr>
<?php
          }
 ?>
</table>
 </td>
     </tr>
   </table>
        <table width="200" border="1" cellpadding="1" cellspacing="0"
bordercolor="#000000" align="center">
     <tr>
```

```
    <td>
      <table width="478" border="0" bgcolor="#E9E9E9">
   <tr>
    <td width="472"><strong>VISITAS POR NAVEGADOR </strong></td>
   </tr>
  </table>
      <table width="478" border="0" >
<tr>
  <td width="161" bgcolor="#CCCCCC"><strong>Navegador </strong></td>
  <td width="115" bgcolor="#CCCCCC"><strong>Paginas Visitas </strong></td>
  <td width="188" bgcolor="#CCCCCC"><strong>Visitas</strong></td>
</tr>
<?php
// presenta los trafico segun el navegador
  $reg=count($data1['data']);
      $r=$data1['data'];
      for($i=0;$i<$reg;$i++){
        $fila=$r[$i];
?>
<tr>
```

```
    <td><?php echo $fila['browser']; ?></td>
    <td><?php echo $fila['pageviews']; ?></td>
    <td><?php echo $fila['visits']; ?></td>
  </tr>
 <?php
            }
  ?>
</table>
</td>
      </tr>
    </table>
<?php
include("grafico.php");
?>
          <table width="200" border="1" cellpadding="1" cellspacing="0"
bordercolor="#000000" align="center">
      <tr>
        <td>
          <table width="478" border="0" bgcolor="#E9E9E9">
      <tr>
```

```
    <td width="472"><strong>PAGINAS VISITADAS </strong></td>
   </tr>
  </table>
        <table width="478" border="0" >
<tr>
  <td width="400" bgcolor="#CCCCCC"><strong>Pagina </strong></td>
  <td width="78" bgcolor="#CCCCCC"><strong>Visitas</strong></td>
</tr>
<?php
// presenta los trafico segun las paginas visitas
  $reg=count($data2['data']);
        $r=$data2['data'];
        for($i=0;$i<$reg;$i++){
          $fila=$r[$i];

?>
<tr>
  <td><?php echo $fila['pagePath']; ?></td>
  <td><?php echo $fila['visits']; ?></td>
</tr>
```

```
 <?php
          }
  ?>
</table>
</td>
     </tr>
   </table>
<?php
// Libreria para presentar grafico estadistico
include "function_dates.php"; include "googleanalytics.class.php";
$fecha2 = date('Y-m-d');
$fecha1 = strtotime ( '-15 day' , strtotime ( $fecha2 ) ) ;
$fecha1 = date ( 'Y-m-d' , $fecha1 );
$fechas = createDateRangeArray($fecha1,$fecha2);
$days = count($fechas);

try {
          $ga = new GoogleAnalytics('fernancarpio@gmail.com','nando2');
          $ga->setProfile('ga:59231567');
       $ga->setDateRange($fecha1,$fecha2);
```

```
        $report = $ga->getReport(


         array('dimensions'=>urlencode('ga:date'),

         'metrics'=>urlencode('ga:visits'),

          null,

        'sort'=>'ga:date'

                                        )

                          );
$i = 0; $serie_dim = null; $serie_val = null;

foreach ($report as $valor){

    $serie_dim .= """.$fechas[$i].'", ";

    $serie_val .= $valor['ga:visits']."; ";

    $i++;


}


$serie_dim = substr($serie_dim, 0, -2); $serie_val = substr($serie_val, 0, -2);

//presenta grafico del trafico
```

```
$script1 = "<script
type='text/javascript'src='http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min
.js'></script><script type='text/javascript'>var
chart;$(document).ready(function(){chart=new
Highcharts.Chart({chart:{renderTo:'container',defaultSeriesType:'line',marginRight:1
30,marginBottom:25},title:{text:'VISITAS POR DIA (Ultimos 15 dias)',x:-
20},subtitle:{text:'Fuente: Google Analytics',x:-20},xAxis:{categories:['.

$serie_dim.

']},yAxis:{title:{text:'Visitas'},plotLines:[{value:0,width:1,color:'#808080'}]},tooltip
:{formatter:function(){return'<b>'+this.series.name+'</b><br/>'+

this.x+': '+this.y+';}},legend:{layout:'vertical',align:'right',verticalAlign:'top',x:-
10,y:100,borderWidth:0},series:[{name:'Visitas',data:['.

$serie_val.

']}]});});</script><script
type='text/javascript'src='http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min
.js'></script><script type='text/javascript'src='highcharts.js'></script><script
type='text/javascript'src='exporting.js'></script><div id='container'style='width:
600px; height: 400px; margin: 0 auto'></div>";


}catch (Exception $e) {

        print 'Error: ' . $e->getMessage();

}

echo $script1;
```

```
?>
<?php
// include the Google Analytics PHP class
include "googleanalytics.class.php";
try {
	// create an instance of the GoogleAnalytics class using your own Google {email} and {password}
	$ga = new GoogleAnalytics('{email}','{password}');

	// set the Google Analytics profile you want to access - format is 'ga:123456';
	$ga->setProfile('{GA Profile ID}');

	// set the date range we want for the report - format is YYYY-MM-DD
	$ga->setDateRange('2009-04-01','2009-04-07');

	// get the report for date and country filtered by Australia, showing pageviews and visits
	$report = $ga->getReport(
		array('dimensions'=>urlencode('ga:date,ga:country'),
			'metrics'=>urlencode('ga:pageviews,ga:visits'),
			'filters'=>urlencode('ga:country=@Australia'),
```

```
                    'sort'=>'-ga:pageviews'
                    )
                );

        //print out the $report array
        print "<pre>";
        print_r($report);
        print "</pre>";

} catch (Exception $e) {
        print 'Error: ' . $e->getMessage();
}
?>
?php
/**
 * Google Analytics API Class
 *
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
```

```
 * License as published by the Free Software Foundation; either

 * version 2.1 of the License, or (at your option) any later version.

 *

 * This library is distributed in the hope that it will be useful,

 * but WITHOUT ANY WARRANTY; without even the implied warranty of

 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU

 * Lesser General Public License for more details.

 *

 * You should have received a copy of the GNU Lesser General Public

 * License along with this library; if not, write to the Free Software

 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

 *

 * Author: Doug Tan

 * Website: http://www.askaboutphp.com/

 * Version: 0.2

 */


class GoogleAnalytics {
```

```
        private $_email;
        private $_passwd;
        private $_authCode;
        private $_profileId;

        private $_endDate;
        private $_startDate;

        /**
* public constructor
*
* @param string $email
* @param string $passwd
* @return class
*/
        public function __construct($email, $passwd) {
                // set the email and password
                $this->_email = $email;
                $this->_passwd = $passwd;
```

```
                //default the start and end date
                $this->_endDate = date('Y-m-d', mktime(0, 0, 0, date("m") , date("d") -
1, date("Y")));
                $this->_startDate = date('Y-m-d', mktime(0, 0, 0, date("m") , date("d")
- 31, date("Y")));

                if (isset($this->_email) && isset($this->_passwd)) {
                        // autenticate the user
                        if (!$this->_authenticate()) {
                                //print $this->_authCode;
                                throw new Exception('Failed to authenticate, please
check your email and password.');
                        }
                }
        }

        /**
   * Sets Profile ID
        *
   * @param string $id (format: 'ga:1234')
   */
```

```
        public function setProfile($id) {
                //look for a match for the pattern ga:XXXXXXXX, of up to 10 digits
                if (!preg_match('/^ga:\d{1,10}/',$id)) {
                        throw new Exception('Invalid GA Profile ID set. The format should ga:XXXXXX, where XXXXXX is your profile number');
                }
                $this->_profileId = $id;
                return TRUE;
        }

        /**
    * Sets the date range
    *
    * @param string $startDate (YYYY-MM-DD)
    * @param string $endDate   (YYYY-MM-DD)
    */
        public function setDateRange($startDate, $endDate) {
                //validate the dates
                if (!preg_match('/\d{4}-\d{2}-\d{2}/', $startDate)) {
                        throw new Exception('Format for start date is wrong, expecting YYYY-MM-DD format');
```

```
                }
                if (!preg_match('/\d{4}-\d{2}-\d{2}/', $endDate)) {
                        throw new Exception('Format for end date is wrong, expecting
YYYY-MM-DD format');
                }
                if (strtotime($startDate)>strtotime($endDate)) {
                        throw new Exception('Invalid Date Range. Start Date is greated
than End Date');
                }
                $this->_startDate = $startDate;
                $this->_endDate = $endDate;
                return TRUE;
        }

        /**
   * Retrieve the report according to the properties set in $properties
        *
   * @param array $properties
        * @return array
   */
        public function getReport($properties = array()) {
```

```
                if (!count($properties)) {
                        die ('getReport requires valid parameter to be passed');
                        return FALSE;
                }

                //arrange the properties in key-value pairing
                foreach($properties as $key => $value){
      $params[] = $key.'='.$value;
    }
                //compose the apiURL string
    $apiUrl = 'https://www.google.com/analytics/feeds/data?ids='.$this->_profileId.'&start-date='.$this->_startDate.'&end-date='.$this->_endDate.'&'.implode('&', $params);

                //call the API
                $xml = $this->_callAPI($apiUrl);
                //get the results
                if ($xml) {
                        $dom = new DOMDocument();
                        $dom->loadXML($xml);
                        $entries = $dom->getElementsByTagName('entry');
```

```
                foreach ($entries as $entry){
                        $dimensions = $entry-
>getElementsByTagName('dimension');
                        foreach ($dimensions as $dimension) {
                                $dims .= $dimension->getAttribute('value').'~~';
                        }

                        $metrics = $entry->getElementsByTagName('metric');
                        foreach ($metrics as $metric) {
                                $name = $metric->getAttribute('name');
                                $mets[$name] = $metric->getAttribute('value');
                        }

                        $dims = trim($dims,'~~');
                        $results[$dims] = $mets;

                        $dims='';
                        $mets='';
                }
        } else {
```

```
                        throw new Exception('getReport() failed to get a valid XML
from Google Analytics API service');
                }
                return $results;
        }

        /**
   * Retrieve the list of Website Profiles according to your GA account
        *
   * @param none
        * @return array
   */
        public function getWebsiteProfiles() {

                // make the call to the API
                $response = $this-
>_callAPI('https://www.google.com/analytics/feeds/accounts/default');

                //parse the response from the API using DOMDocument.
                if ($response) {
                        $dom = new DOMDocument();
```

```
                $dom->loadXML($response);
                $entries = $dom->getElementsByTagName('entry');
                foreach($entries as $entry){
                        $tmp['title'] = $entry->getElementsByTagName('title')-
>item(0)->nodeValue;
                        $tmp['id'] = $entry->getElementsByTagName('id')-
>item(0)->nodeValue;
                        foreach($entry->getElementsByTagName('property') as
$property){
                                if (strcmp($property->getAttribute('name'),
'ga:accountId') == 0){
                                        $tmp["accountId"] = $property-
>getAttribute('value');
                                }
                                if (strcmp($property->getAttribute('name'),
'ga:accountName') == 0){
                                  $tmp["accountName"] = $property-
>getAttribute('value');
                                }
                                if (strcmp($property->getAttribute('name'),
'ga:profileId') == 0){
                                        $tmp["profileId"] = $property-
>getAttribute('value');
```

```
                                        }
                                        if (strcmp($property->getAttribute('name'),
'ga:webPropertyId') == 0){
                                                $tmp["webProfileId"] = $property-
>getAttribute('value');
                                        }
                                }
                                $profiles[] = $tmp;
                        }
                } else {
                        throw new Exception('getWebsiteProfiles() failed to get a valid
XML from Google Analytics API service');
                }
                return $profiles;
        }

        /**
   * Make the API call to the $url with the $_authCode specified
        *
   * @param url
        * @return result from _postTo
```

```
   */
        private function _callAPI($url) {
                return $this->_postTo($url,array(),array("Authorization: GoogleLogin
auth=".$this->_authCode));
        }

        /**
   * Authenticate the email and password with Google, and set the $_authCode return
by Google
        *
   * @param none
        * @return none
   */
        private function _authenticate() {
                $postdata = array(
        'accountType' => 'GOOGLE',
        'Email' => $this->_email,
        'Passwd' => $this->_passwd,
        'service' => 'analytics',
        'source' => 'askaboutphp-v01'
      );
```

```
                $response = $this->_postTo("https://www.google.com/accounts/ClientLogin", $postdata);
                //process the response;
                if ($response) {
                        preg_match('/Auth=(.*)/', $response, $matches);
                        if(isset($matches[1])) {
                                $this->_authCode = $matches[1];
                                return TRUE;
                        }
                }
                return FALSE;
        }

        /**
   * Performs the curl calls to the $url specified.
        *
   * @param string $url
        * @param array $data - specify the data to be 'POST'ed to $url
        * @param array $header - specify any header information
```

```
        * @return $response from submission to $url
*/
        private function _postTo($url, $data=array(), $header=array()) {

                //check that the url is provided
                if (!isset($url)) {
                        return FALSE;
                }

                //send the data by curl
                $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
                curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
                if (count($data)>0) {
                        //POST METHOD
                        curl_setopt($ch, CURLOPT_POST, TRUE);
                        curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
                } else {
                        $header[] = array("application/x-www-form-urlencoded");
```

```
                curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
        }

        $response = curl_exec($ch);
$info = curl_getinfo($ch);

curl_close($ch);

        //print_r($info);
        //print $response;
        if($info['http_code'] == 200) {
                return $response;
        } elseif ($info['http_code'] == 400) {
                throw new Exception('Bad request - '.$response);
        } elseif ($info['http_code'] == 401) {
                throw new Exception('Permission Denied - '.$response);
        } else {
                return FALSE;
        }
```

```
        }
}
?>
<?php
function createDateRangeArray($strDateFrom,$strDateTo) {
  // takes two dates formatted as YYYY-MM-DD and creates an
  // inclusive array of the dates between the from and to dates.

  // could test validity of dates here but I'm already doing
  // that in the main script
  $aryRange=array();
  $iDateFrom=mktime(1,0,0,substr($strDateFrom,5,2),
substr($strDateFrom,8,2),substr($strDateFrom,0,4));
  $iDateTo=mktime(1,0,0,substr($strDateTo,5,2),
substr($strDateTo,8,2),substr($strDateTo,0,4));

  if ($iDateTo>=$iDateFrom) {
    array_push($aryRange,date('Y-m-d',$iDateFrom)); // first entry

    while ($iDateFrom<$iDateTo) {
      $iDateFrom+=86400; // add 24 hours
```

```
    array_push($aryRange,date('Y-m-d',$iDateFrom));
   }
  }
 return $aryRange;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de-de" lang="de-de"
dir="ltr" >
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <meta name="generator" content="Joomla! 1.5 - Open Source Content
Management" />
        <title>Uninstall of Olark Chat Module for Joomla 1.5</title>
</head>
<body>
<?php
$tFiles=array();
$tFiles[]="mgnote.php";
$tFiles[]="mgtext.php";
```

```
$tFiles[]="mgspacer.php";

define(DS, "/");

$sTargetPath =
'..'.DS.'..'.DS.'libraries'.DS.'joomla'.DS.'html'.DS.'parameter'.DS.'element';


foreach($tFiles as $sFile)

{

	if(file_exists($sTargetPath.DS.$sFile))

	{

		echo "Uninstall '".$sTargetPath.DS.$sFile."'<br />\n";

		unlink($sTargetPath.DS.$sFile);

	}

	else

		echo "'".$sSourcePath.DS.$sFile."' not found.<br />\n";

}

?>

<br />

Goto module administration <a
href="../../administrator/index.php?option=com_modules">here</a>.

</body>

</html>
```

```
efined('_JEXEC') or die('Direct Access to this location is not allowed.');

$sOlarkId=$params->get('olarkid', '');

$sSayText=(($params->get('say_text_DEFAULT', '0') == '1') ? JText::_("Type here and hit <enter> to chat") : $params->get('say_text', ''));

if($sSayText == "Type here and hit <enter> to chat")

	$sSayText="";


$sNotAvailableText=(($params->get('not_available_text_DEFAULT', '0') == '1') ? JText::_("Live Help: Offline") : $params->get('not_available_text', ''));

if($sNotAvailableText == "Live Help: Offline")

	$sNotAvailableText="";

$sOfflineMessage=(($params->get('offline_message_DEFAULT', '0') == '1') ? JText::_("Our live support feature is currently offline. Please try again later.") : $params->get('offline_message', ''));

if($sOfflineMessage == "Our live support feature is currently offline. Please try again later.")

	$sOfflineMessage="";

$sBusyText=(($params->get('busy_text_DEFAULT', '0') == '1') ? JText::_("Live Help: Unavailable") : $params->get('busy_text', ''));

if($sBusyText == "Live Help: Unavailable")

	$sBusyText="";
```

```
$sBusyMessage=(($params->get('busy_message_DEFAULT', '0') == '1') ?
JText::_("All of our representatives are with other customers at this time. We will be with you shortly.") : $params->get('busy_message', ''));

if($sBusyMessage == "All of our representatives are with other customers at this time. We will be with you shortly.")

        $sBusyMessage="";


$sBeforeChatText=(($params->get('before_chat_text_DEFAULT', '0') == '1') ?
JText::_("Click for Live Help") : $params->get('before_chat_text', ''));

if($sBeforeChatText == "Click for Live Help")

        $sBeforeChatText="";

$sInChatText=(($params->get('in_chat_text_DEFAULT', '0') == '1') ? JText::_("Live Help: Now Chatting") : $params->get('in_chat_text', ''));

if($sInChatText == "Live Help: Now Chatting")

        $sInChatText="";

$sWelcomeMsg=(($params->get('welcome_msg_DEFAULT', '0') == '1') ?
JText::_("Questions? This window is your portal to live online support. We are around, send us a message.") : $params->get('welcome_msg', ''));

if($sWelcomeMsg == "Questions? This window is your portal to live online support. We are around, send us a message.")

        $sWelcomeMsg="";
```

```
$sPreChatMessage=(($params->get('pre_chat_message_DEFAULT', '0') == '1') ? JText::_("Hi, I am around, click 'start chatting' to contact me.") : $params->get('pre_chat_message', ''));

if($sPreChatMessage == "Hi, I am around, click 'start chatting' to contact me.")

	$sPreChatMessage="";

$sPreChatErrorText=(($params->get('pre_chat_error_text_DEFAULT', '0') == '1') ? JText::_("Please enter your name and email in case we get disconnected.") : $params->get('pre_chat_error_text', ''));

if($sPreChatErrorText == "Please enter your name and email in case we get disconnected.")

	$sPreChatErrorText="";

$sPreChatSubmit=(($params->get('pre_chat_submit_DEFAULT', '0') == '1') ? JText::_("Click here to start chatting") : $params->get('pre_chat_submit', ''));

if($sPreChatSubmit == "Click here to start chatting")

	$sPreChatSubmit="";


$sEmailBodyErrorText=(($params->get('email_body_error_text_DEFAULT', '0') == '1') ? JText::_("Complete all fields") : $params->get('email_body_error_text', ''));

if($sEmailBodyErrorText == "Complete all fields")

	$sEmailBodyErrorText="";

$sHablaNameInputText=(($params->get('habla_name_input_text_DEFAULT', '0') == '1') ? JText::_("<click here> and type your Name") : $params->get('habla_name_input_text', ''));
```

```
if($sHablaNameInputText == "<click here> and type your Name")

        $sHablaNameInputText="";

$sHablaOfflineEmailText=(($params->get('habla_offline_email_text_DEFAULT',
'0') == '1') ? JText::_("<click here> and type your Email") : $params-
>get('habla_offline_email_text', ''));

if($sHablaOfflineEmailText == "<click here> and type your Email")

        $sHablaOfflineEmailText="";

$sHablaOfflineSubmitValue=(($params-
>get('habla_offline_submit_value_DEFAULT', '0') == '1') ? JText::_("Send") :
$params->get('habla_offline_submit_value', ''));

if($sHablaOfflineSubmitValue == "Send")

        $sHablaOfflineSubmitValue="";

$sHablaOfflineBodyText=(($params->get('habla_offline_body_text_DEFAULT', '0')
== '1') ? JText::_("We are offline, send us a message") : $params-
>get('habla_offline_body_text', ''));

if($sHablaOfflineBodyText == "We are offline, send us a message")

        $sHablaOfflineBodyText="";

$sHablaOfflineSentText=(($params->get('habla_offline_sent_text_DEFAULT', '0')
== '1') ? JText::_("Thank you for your message. We will get back to you as soon as
we can.") : $params->get('habla_offline_sent_text', ''));

if($sHablaOfflineSentText == "Thank you for your message. We will get back to you
as soon as we can.")

        $sHablaOfflineSentText="";
```

```
$sAwayText=(($params->get('away_text_DEFAULT', '0') == '1') ? JText::_("Live Help: Unavailable") : $params->get('away_text', ''));

if($sAwayText == "Live Help: Unavailable")

	$sAwayText="";

$sSendText=(($params->get('send_text_DEFAULT', '0') == '1') ? JText::_("Send") : $params->get('send_text', ''));

if($sSendText == "Send")

	$sSendText="";

$sSendingText=(($params->get('sending_text_DEFAULT', '0') == '1') ? JText::_("sending") : $params->get('sending_text', ''));

if($sSendingText == "sending")

	$sSendingText="";


$nInline=$params->get('inline', '-1');


$nShowAway=$params->get('show_away', '-1');

$nShowAwayAsHeader=$params->get('show_away_as_header', '-1');


$nHideMinMaxButtons=$params->get('hide_min_max_buttons', '-1');

$nEnableButtons=$params->get('enable_buttons', '-1');

$nParseLinks=$params->get('parse_links', '-1');
```

```
$sUrlHandlerTargetWindow=(($params->get('url_handler_target_window_DEFAULT', '0') == '1') ? "_top" : $params->get('url_handler_target_window', ''));

if($sUrlHandlerTargetWindow == "_top")

	$sUrlHandlerTargetWindow="";

$sUrlLocalPattern=(($params->get('url_local_pattern_DEFAULT', '0') == '1') ? "document.domain" : $params->get('url_local_pattern', ''));

if($sUrlLocalPattern == "document.domain")

	$sUrlLocalPattern="";

$sUrlHandler=(($params->get('url_handler_DEFAULT', '0') == '1') ? "http://static.hab.la/js/html/url_handler.html" : $params->get('url_handler', ''));

if($sUrlHandler == "http://static.hab.la/js/html/url_handler.html")

	$sUrlHandler="";


$sDefaultFlashOffIcon=(($params->get('default_flash_off_icon_DEFAULT', '0') == '1') ? "http://static.hab.la/js/images/white.ico" : $params->get('default_flash_off_icon', ''));

if($sDefaultFlashOffIcon == "http://static.hab.la/js/images/white.ico")

	$sDefaultFlashOffIcon="";

$sDefaultFlashOnIcon=(($params->get('default_flash_on_icon_DEFAULT', '0') == '1') ? "http://static.hab.la/js/images/orange.ico" : $params->get('default_flash_on_icon', ''));

if($sDefaultFlashOnIcon == "http://static.hab.la/js/images/orange.ico")
```

```
        $sDefaultFlashOnIcon="";
$nFlashIcons=$params->get('flash_icons', '-1');

$nDisableGoogleAnalytics=$params->get('disableGoogleAnalytics', '-1');

$bNotSecure=strpos($_SERVER['SERVER_PROTOCOL'], "HTTPS") === false;
?>
<script type="text/javascript" src="http<?php if(!$bNotSecure) echo "s";
?>://static.olark.com/js/wc.js"></script>
<script type="text/javascript">
tConfig=wc_config();
<?php
if($sSayText != "")
        echo "tConfig.vars[\"say_text\"]=\"$sSayText\";\n";

if($sNotAvailableText != "")
        echo "tConfig.vars[\"not_available_text\"]=\"$sNotAvailableText\";\n";
if($sOfflineMessage != "")
        echo "tConfig.vars[\"offline_message\"]=\"$sOfflineMessage\";\n";
if($sBusyText != "")
```

```
        echo "tConfig.vars[\"busy_text\"]=\"$sBusyText\";\n";
if($sBusyMessage != "")
        echo "tConfig.vars[\"busy_message\"]=\"$sBusyMessage\";\n";

if($sBeforeChatText != "")
        echo "tConfig.vars[\"before_chat_text\"]=\"$sBeforeChatText\";\n";
if($sInChatText != "")
        echo "tConfig.vars[\"in_chat_text\"]=\"$sInChatText\";\n";
if($sWelcomeMsg != "")
        echo "tConfig.vars[\"welcome_msg\"]=\"$sWelcomeMsg\";\n";

if($sPreChatMessage != "")
        echo "tConfig.vars[\"pre_chat_message\"]=\"$sPreChatMessage\";\n";
if($sPreChatErrorText != "")
        echo "tConfig.vars[\"pre_chat_error_text\"]=\"$sPreChatErrorText\";\n";
if($sPreChatSubmit != "")
        echo "tConfig.vars[\"pre_chat_submit\"]=\"$sPreChatSubmit\";\n";

if($sEmailBodyErrorText != "")
```

```
        echo
"tConfig.vars[\"email_body_error_text\"]=\"$sEmailBodyErrorText\";\n";

if($sHablaNameInputText != "")

        echo
"tConfig.vars[\"habla_name_input_text\"]=\"$sHablaNameInputText\";\n";

if($sHablaOfflineBodyText != "")

        echo
"tConfig.vars[\"habla_offline_body_text\"]=\"$sHablaOfflineBodyText\";\n";

if($sHablaOfflineEmailText != "")

        echo
"tConfig.vars[\"habla_offline_email_text\"]=\"$sHablaOfflineEmailText\";\n";

if($sHablaOfflineSentText != "")

        echo
"tConfig.vars[\"habla_offline_sent_text\"]=\"$sHablaOfflineSentText\";\n";

if($sHablaOfflineSubmitValue != "")

        echo
"tConfig.vars[\"habla_offline_submit_value\"]=\"$sHablaOfflineSubmitValue\";\n";


if($sAwayText != "")

        echo "tConfig.vars[\"away_text\"]=\"$sAwayText\";\n";

if($sSendText != "")

        echo "tConfig.vars[\"send_text\"]=\"$sSendText\";\n";
```

```
if($sSendingText != "")
	echo "tConfig.vars[\"sending_text\"]=\"$sSendingText\";\n";

if($nInline != "-1")
	echo "tConfig.setInline($nInline);\n";

if($nShowAway != "-1")
	echo "tConfig.vars[\"show_away\"]=$nShowAway;\n";
if($nShowAwayAsHeader != "-1")
	echo "tConfig.vars[\"show_away_as_header\"]=$nShowAwayAsHeader;\n";

if($nHideMinMaxButtons != "-1")
	echo "tConfig.vars[\"hide_min_max_buttons\"]=$nHideMinMaxButtons;\n";
if($nEnableButtons != "-1")
	echo "tConfig.vars[\"enable_buttons\"]=$nEnableButtons;\n";
if($nParseLinks != "-1")
	echo "tConfig.vars[\"parse_links\"]=$nParseLinks;\n";
if($sUrlHandlerTargetWindow != "")
	echo
"tConfig.vars[\"url_handler_target_window\"]=\"$sUrlHandlerTargetWindow\";\n";
```

```
if($sUrlLocalPattern != "")
	echo "tConfig.vars[\"url_local_pattern\"]=\"$sUrlLocalPattern\";\n";
if($sUrlHandler != "")
	echo "tConfig.vars[\"url_handler\"]=\"$sUrlHandler\";\n";


if($sDefaultFlashOffIcon != "")
	echo
"tConfig.vars[\"default_flash_off_icon\"]=\"$sDefaultFlashOffIcon\";\n";
if($sDefaultFlashOnIcon != "")
	echo "tConfig.vars[\"default_flash_on_icon\"]=\"$sDefaultFlashOnIcon\";\n";
if($nFlashIcons != "-1")
	echo "tConfig.vars[\"flash_icons\"]=$nFlashIcons;\n";


if($nDisableGoogleAnalytics != "-1")
	echo
"tConfig.vars[\"disableGoogleAnalytics\"]=$nDisableGoogleAnalytics;\n";
?>
wc_init("<?php echo $sOlarkId; ?>", tConfig);
</script>
class JElementMGText extends JElement
{
```

```
        /**
        * Element name
        *
        * @access       protected
        * @var          string
        */
        var      $_name = 'MGText';
        //--------------------------------------------------------------------------------------------
---------------
        function fetchElement($aName, $aValue, &$aNode, $aControlName)
        {
                $nSize=( $aNode->attributes('size') ? 'size="'.$aNode-
>attributes('size').'"' : '' );
                $sClass=( $aNode->attributes('class') ? 'class="'.$aNode-
>attributes('class').'"' : 'class="text_area"' );
                $sDefault=$aNode->attributes('default');

                if($aValue == "" || trim($aValue) == trim($sDefault))
                {
                        $sChecked='checked="checked"';
```

```
                $aValue=( $aNode->attributes('default') ? $aNode->attributes('default') : '' );
                $sDisabled='disabled="disabled"';
        }
        else
        {
                $sChecked="";
        }

        $sTitle=($aNode->attributes('title') ? JText::_($aNode->attributes('title')) : $aValue);

        /*
        * Required to avoid a cycle of encoding &
        * html_entity_decode was used in place of htmlspecialchars_decode because
        * htmlspecialchars_decode is not compatible with PHP 4
        */
        $aValue=htmlspecialchars(html_entity_decode($aValue, ENT_QUOTES), ENT_QUOTES);
```

```
                $sOnClickChkBox= "onclick=\"";

                $sOnClickChkBox.="(this.checked ";

                $sOnClickChkBox.="?
document.getElementById('$aControlName$aName').value='".addslashes($sDefault).
"' ";

                $sOnClickChkBox.=":
document.getElementById('$aControlName$aName').value=''); ";

                $sOnClickChkBox.="(this.checked ";

                $sOnClickChkBox.="?
document.getElementById('$aControlName$aName').disabled=true ";

                $sOnClickChkBox.=":
document.getElementById('$aControlName$aName').disabled=false);\"";

                return '<input type="checkbox"
name="'.$aControlName.'['.$aName.'_DEFAULT]" value="1" '.$sChecked.'
'.$sOnClickChkBox.' />'.JText::_("Default")

                        .' <input type="text" name="'.$aControlName.'['.$aName.']"
id="'.$aControlName.$aName.'" value="'.$aValue.'" title="'.$sTitle.'" alt="'.$sTitle.'"
'.$sClass.' '.$sSize.' '.$sDisabled.' />';

        }

}

defined('JPATH_BASE') or die();
```

```
/**
 * Renders a spacer element
 *
 * @package   Joomla.Framework
 * @subpackage        Parameter
 * @since             1.5
 */
class JElementMGSpacer extends JElement
{
        /**
        * Element name
        *
        * @access       protected
        * @var          string
        */
        var      $_name = 'MGSpacer';


//-----------------------------------------------------------------------------------------------

        function fetchTooltip($aLabel, $aDescription, &$aNode, $aControlName,
$aName)
```

```
        {
                return $aDescription;
        }
//-------------------------------------------------------------------------------------------------
        function fetchElement($aName, $aValue, &$aNode, $aControlName)
        {
                $sStyle1=( $aNode->attributes('style') ? 'style="'.$aNode-
>attributes('style').'"' : 'style="font-weight: bold; text-align: center;"' );
                $sStyle2=( $aNode->attributes('style2') ? 'style="'.$aNode-
>attributes('style2').'"' : 'style="margin-top: -15px;"' );
                $sStyle3=( $aNode->attributes('style3') ? 'style="'.$aNode-
>attributes('style3').'"' : 'style="background-color: #F6F6F6; padding-left: 5px;
padding-right: 5px;"' );
                $sClass=( $aNode->attributes('class') ? 'class="'.$aNode-
>attributes('class').'"' : 'class="mgspacer"' );

                return "<div $sClass $sStyle1><hr /><div $sStyle2><span
$sStyle3>".JText::_($aValue)."</span></div></div>";
        }
}
defined('JPATH_BASE') or die();
```

```
//----------------------------------------------------------------------------------------------

/**
 * Renders a note element
 *
 * @package   Joomla.Framework
 * @subpackage        Parameter
 * @since             1.5
 */
class JElementMGNote extends JElement
{
        /**
        * Element name
        *
        * @access      protected
        * @var         string
        */
        var      $_name = 'MGNode';
        //----------------------------------------------------------------------------------------------
        function fetchElement($aName, $aValue, &$aNode, $aControlName)
        {
```

```
                $sStyle=( $aNode->attributes('style') ? 'style="'.$aNode->attributes('style').'"' : '' );
                $sClass=( $aNode->attributes('class') ? 'class="'.$aNode->attributes('class').'"' : 'class="mgnote"' );

                return "<span $sClass $sStyle>".JText::_($aValue)."</span>";
        }
}
defined('_JEXEC') or die('Direct Access to this location is not allowed.');

class modFeedHelper
{
        function getFeed($params)
        {
                // module params
                $rssurl = $params->get('rssurl', '');

                //  get RSS parsed object
                $options = array();
                $options['rssUrl']              = $rssurl;
                if ($params->get('cache')) {
```

```
        $options['cache_time']  = $params->get('cache_time', 15) ;
        $options['cache_time']          *= 60;
} else {
        $options['cache_time'] = null;
}

$rssDoc =& JFactory::getXMLparser('RSS', $options);

$feed = new stdclass();

if ($rssDoc != false)
{
        // channel header and link
        $feed->title = $rssDoc->get_title();
        $feed->link = $rssDoc->get_link();
        $feed->description = $rssDoc->get_description();

        // channel image if exists
        $feed->image->url = $rssDoc->get_image_url();
        $feed->image->title = $rssDoc->get_image_title();
```

```
                    // items
                    $items = $rssDoc->get_items();

                    // feed elements
                    $feed->items = array_slice($items, 0, $params->get('rssitems',
5));
                } else {
                    $feed = false;
                }

                return $feed;
        }
}
```

MANUAL DE USUARIO

CAPÍTULO II

MANUAL USUARIO

2.1 INTRODUCCIÓN

Este manual contiene información respecto a cómo operar el sitio web, además indicaciones de cómo funciona y los conocimientos mínimos que debe tener el usuario para transmitir y comprender la idea del autor.

2.2 OBJETIVOS DE ESTE MANUAL

Ayudar al usuario en general al correcto desenvolvimiento en nuestro sitio web, el mismo que comprende:

- ✓ Conocer el alcance de todo el sitio por medio de una explicación detallada e ilustrada de cada una de las opciones que lo forman.
- ✓ Funciones no previstas o inesperadas.

2.3 A QUIEN VA DIRIGIDO ESTE MANUAL

Está dirigido, a todos los cibernautas que deseen interactuar con un Mashup de contenido, así como a todos aquellos que deseen explotar las bondades del sitio para

aprovechar de una manera eficaz el desempeño de las Tecnologías de Información y mejorar la experiencia del usuario.

2.4 CONOCIMIENTOS PREVIOS

Se recomienda, que los usuarios tengan conocimientos generales de cómo buscar información en Internet, el saber utilizar los navegadores que existen en el mercado. Ya que la usabilidad del sitio está calificada para resolverlo bajo cualquier servidor, Sistema Operativo e Idioma del sitio. Es decir estamos aptos para que la Interacción Persona computador, sea una experiencia cómoda e inolvidable, lo que le permite regresar al sitio. Y quien quita ser una fuente de ingresos colocando publicidad.

2.5 ACERCA DE ESTE MANUAL

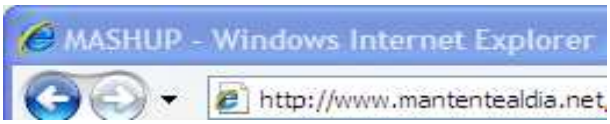
Este manual, contiene las instrucciones necesarias, para indicar cuáles son las mejores opciones a seguir para el desempeño de una funcionalidad en especial.

2.6 CARACTERÍSTICAS DEL SITIO

2.7.1 INTRODUCCIÓN

En un navegador, colocamos la url:

Gráfico No. 24 Url Oficial del Mashup



Elaborado por: Araceli Dávila M.

El equivalente para Mozilla Firefox, Konqueror, bajo Windows, Bajo Linux, Bajo Mac, Bajo una conexión ADSL, Dial Up e inclusive Inalámbrica, el resultado es el mismo.

2.7.2 BENEFICIOS

- ✓ Queremos contribuir con una nueva forma de interactuar con su clientela, dando a conocer sus servicios, noticias de sus productos.
- ✓ Al tener un site alojado en Internet, trabaja las 24 horas, por 7 días a la semana. Teniendo disponibilidad 100% del canal a sus clientes.
- ✓ Universalidad de la web, al tener acceso desde un celular o su Pc, Tablet, etc.

- ✓ Contribuimos con tener espacio en la web, lo que mejora en la imagen de su empresa y posicionamiento de una marca.

2.8 MASHUP DE CONTENIDOS

2.8.1 PÁGINA PRINCIPAL

Gráfico No. 25 Página principal www.mantentealdia.net

Elaborado por: Araceli Dávila M.

2.9 DESCRIPCIÓN DEL SITIO

2.9.1 ADMINISTRACIÓN

Todos los sitios Joomla, cuentan con un manejador de “Catalogos”(por llamarlo así), a las diferentes características que se manejarán en el sitio. El link para acceder, es este:

Gráfico No. 26 Sitio Web de Administración

Elaborado por: Araceli Dávila M.

2.9.1.1 Inicio de Sesión

La página de acceso, enseguida nos llevará a logearnos con nuestro user y password.

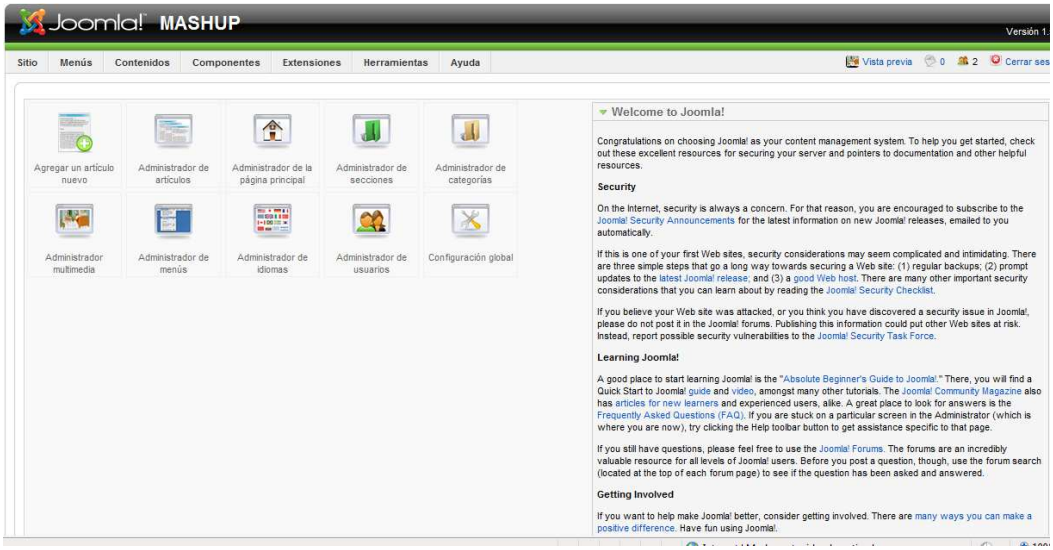
Gráfico No. 27 Loggin de Inicio – Administración Joomla

Iniciar sesión en la administración de Joomla!
Use un nombre de usuario y contraseña válidos para acceder al área de administración.
Volver a la página de inicio del sitio
Nombre de usuario
Contraseña
Idioma
Predet.
Acceder

Elaborado por: Araceli Dávila M.

Cómo se indicó previamente, Joomla maneja como Catálogos: Los artículos, secciones, categorías, contenido multimedia, componentes, pluggins, usuarios, esto es claro en la pantalla principal de Administración.

Gráfico No. 28 Pantalla principal de Administración

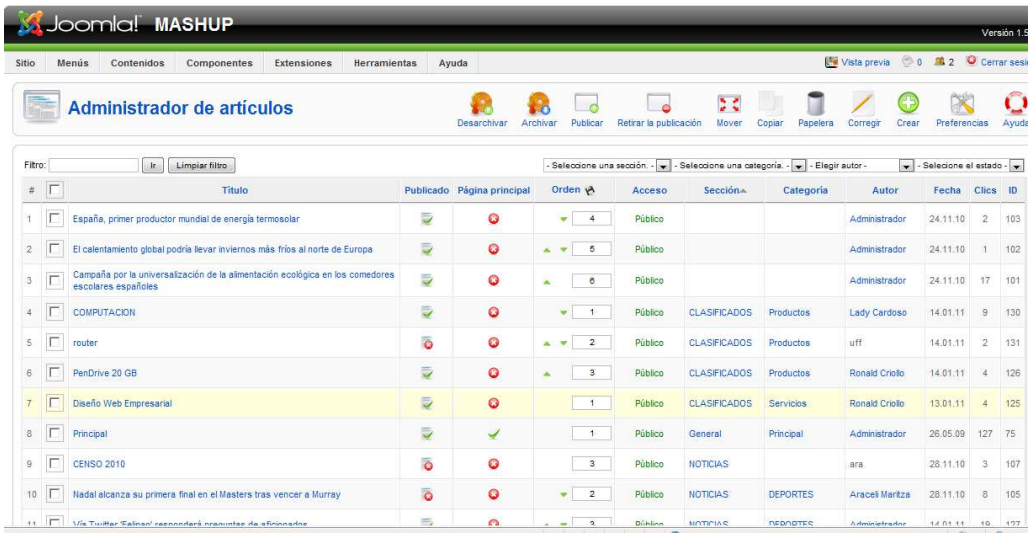

Joomla! MASHUP
Sitio
Menús
Contenidos
Componentes
Extensiones
Herramientas
Ayuda
Vista previa
Cerrar ses
Agregar un artículo nuevo
Administrador de artículos
Administrador de la página principal
Administrador de secciones
Administrador de categorías
Administrador multimedia
Administrador de menús
Administrador de idiomas
Administrador de usuarios
Configuración global
Welcome to Joomla!
Congratulations on choosing Joomla! as your content management system. To help you get started, check out these excellent resources for securing your server and pointers to documentation and other helpful resources.
Security
On the Internet, security is always a concern. For that reason, you are encouraged to subscribe to the Joomla! Security Announcements for the latest information on new Joomla! releases, emailed to you automatically.
If this is one of your first Web sites, security considerations may seem complicated and intimidating. There are three simple steps that go a long way towards securing a Web site: (1) regular backups; (2) prompt updates to the latest Joomla! release; and (3) a good Web host. There are many other important security considerations that you can learn about by reading the Joomla! Security Checklist.
If you believe your Web site was attacked, or you think you have discovered a security issue in Joomla!, please do not post it in the Joomla! forums. Publishing this information could put other Web sites at risk. Instead, report possible security vulnerabilities to the Joomla! Security Task Force.
Learning Joomla!
A good place to start learning Joomla! is the "Absolute Beginner's Guide to Joomla!." There, you will find a Quick Start to Joomla! guide and video, amongst many other tutorials. The Joomla! Community Magazine also has articles for new learners and experienced users, alike. A great place to look for answers is the Frequently Asked Questions (FAQ). If you are stuck on a particular screen in the Administrator (which is where you are now), try clicking the Help toolbar button to get assistance specific to that page.
If you still have questions, please feel free to use the Joomla! Forums. The forums are an incredibly valuable resource for all levels of Joomla! users. Before you post a question, though, use the forum search (located at the top of each forum page) to see if the question has been asked and answered.
Getting Involved
If you want to help make Joomla! better, consider getting involved. There are many ways you can make a positive difference. Have fun using Joomla!.

Elaborado por: Araceli Dávila M.

2.9.1.2 Administrador de Artículos

Nuestra materia prima, son los artículos, para lo cual el Administrador de artículos maneja las opciones de crear, publicar una noticia seleccionada, archivar, despublicar.

Gráfico No. 29 Administrador de Artículos



Elaborado por: Araceli Dávila M.

Este gestor, muy intuitivo por cierto; muestra la fecha de creación, id en Base de datos, y si un artículo se publica solo es cuestión de dar click sobre el ícono correspondiente para aplicar.

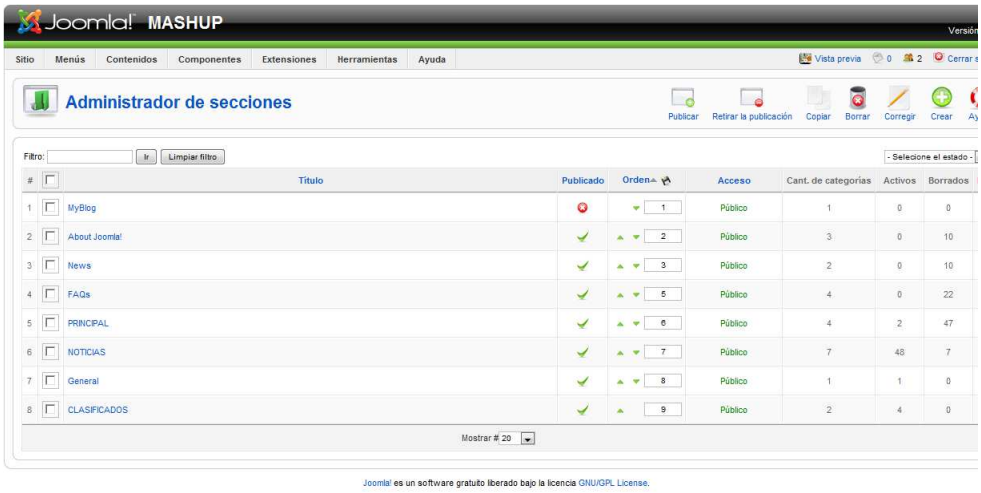
2.9.1.3 Administrador de Secciones

También es necesario para nuestra organización, que los artículos estén asignados en una respectiva Categoría, y estas en una respectiva Sección

El Administrador de Secciones, al igual que el concepto de artículos es muy intuitivo.

Esto quiere decir, que si se despublica toda una sección, se despublican todos los artículos incluidos en esa respectiva categoría y esta a su vez de su sección.

Gráfico No. 30 Administrador de Artículos

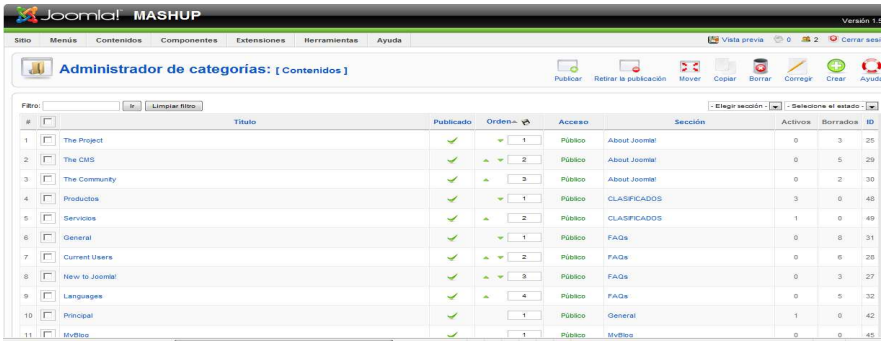


Elaborado por: Araceli Dávila M.

2.9.1.4 Administrador de Categorías

Como ya indicamos en el concepto anterior, las categorías se ubican dentro de una respectiva sección, y las opciones a gestionar son publicar, retiro de publicación, creación.

Gráfico No. 31 Administrador de Categorías



Elaborado por: Araceli Dávila M.

2.9.1.5 Administrador de Idiomas

El Gestor administrativo se encuentra en el menú:

Extensiones->Administrador de idiomas.

El idioma predeterminado, se lo reconocerá por el dibujo de la estrella amarilla, donde nos muestra una lista de paquetes instaladas.

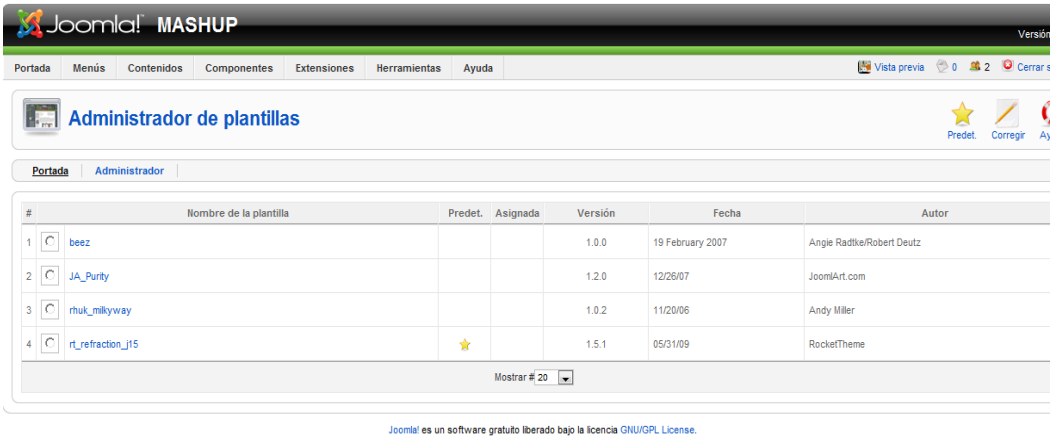
Gráfico No. 32 Administrador de Idiomas

Elaborado por: Araceli Dávila M.

2.9.1.6 Administrador de Plantillas

En este gestor, aparecen las plantillas instaladas en nuestro sitio, cabe destacar que sobre este diseño, se crearan los distintos componentes como Encuestas, Chat, Blogs, Slider que vienen en formatos prediseñados.

Gráfico No. 33 Administrador de Plantillas



Elaborado por: Araceli Dávila M.

Las plantillas, tienen la factibilidad de cambiar los tipos de letras, tamaños, colores, y estilos.

Al colocarnos sobre una plantilla podemos ver una presentación preliminar.

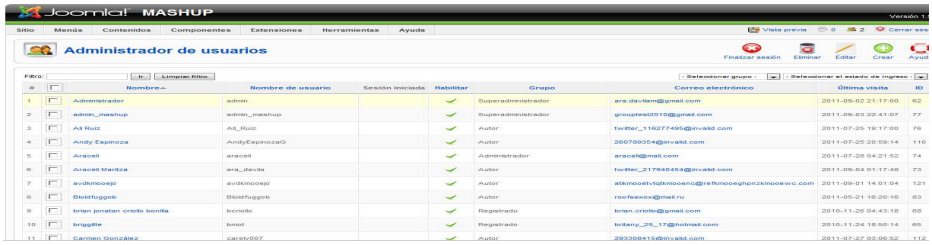
Gráfico No. 34 Presentación preliminares de plantillas

Elaborado por: Araceli Dávila M.

2.9.1.7 Administrador de Usuarios

El Administrador es el siguiente:

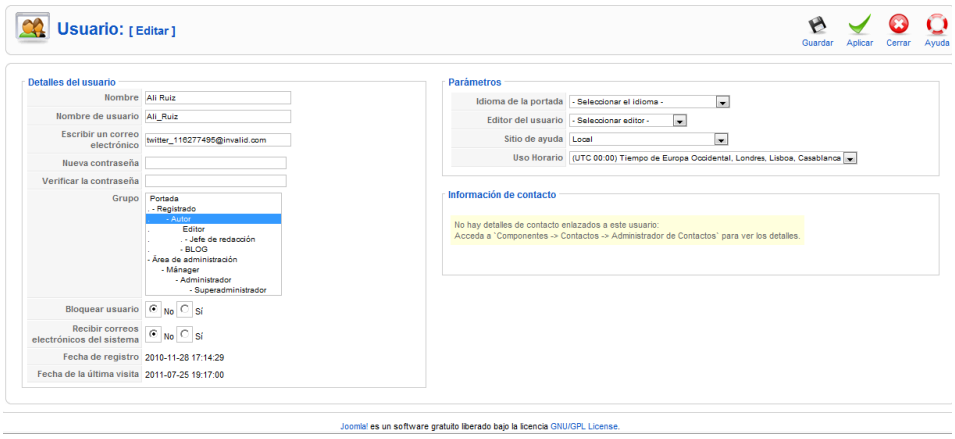
Gráfico No.35 Administrador de Usuarios



Elaborado por: Araceli Dávila M.

Este nos permite gestionar, la creación directa por medio de otro usuario Administrador, Finalizar la sesión de los usuarios conectados al sitio, dar de alta a los usuarios que están registrados, etc.

Gráfico No. 36 Formulario de Ingreso a un usuario

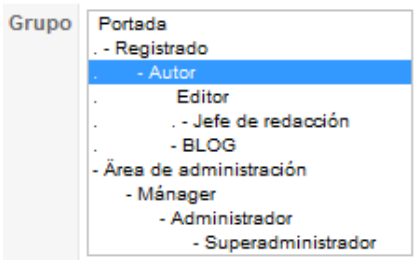


Elaborado por: Araceli Dávila M.

Asi mismo, se puede asignar el idioma de Front End (Portal), asignado a ese usuario, bloquearlo, etc.

Cada usuario tiene un perfil de seguridad, donde nosotros los administradores los podemos ubicar:

Gráfico No. 37 Grupo de Seguridad asignado al usuario



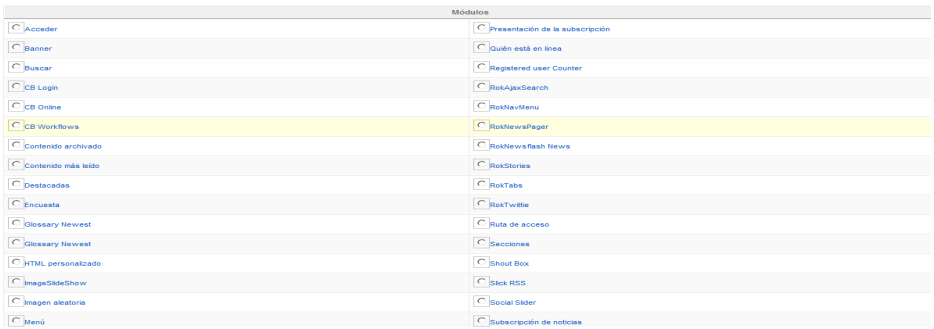
Elaborado por: Araceli Dávila M.

2.9.1.8 Administrador de Módulos

Quizás, el corazón de Joomla, es aquel que maneja, crea y organiza los diferentes módulos que se pueden instalar, para elaborar una funcionalidad “x”.

Los módulos nativos son, Encuestas, Inicio de Sesión, Contactenos, Rss, Pie de Página, Who is Online?, ruta de acceso.

Gráfico No. 38 Módulos



Elaborado por: Araceli Dávila M.

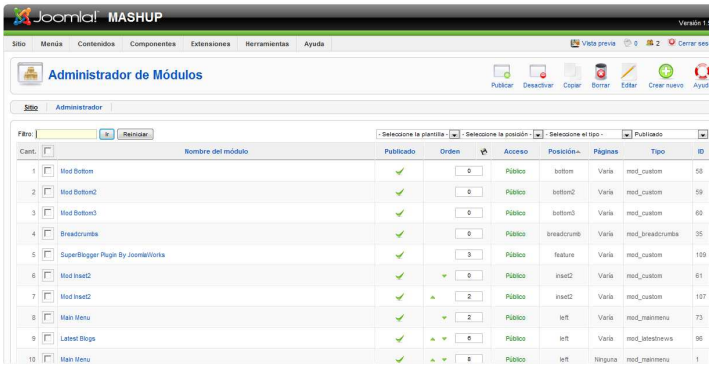
Podemos desatacar a módulos que han sido bajados de Internet, para aplicar una funcionalidad en especial.

Entre ellas el módulo CB Twitter, el cual utiliza la Api de http://twitter.com/ para enlazar nuestra página con el servicio respectivo en conjunto con CB Login.

Asi también, Shout Box(Chat), Slick Rss(Suscriptores de Noticias enlazado a www.eluniverso.com).

El Administrador de módulos es el siguiente:

Gráfico No. 39 Administrador de Módulos



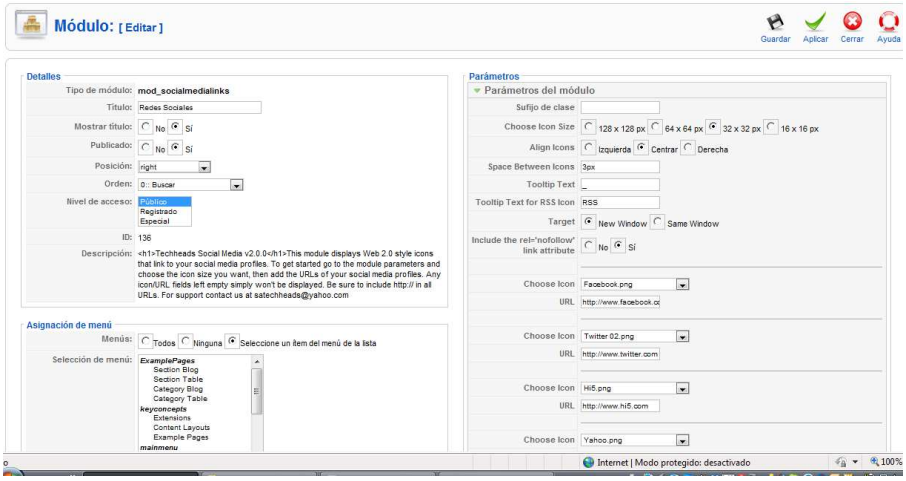
Elaborado por: Araceli Dávila M.

Veamos un poco el módulo de Redes Sociales, para aquello seleccionamos el respectivo ítem:

Gráfico No. 40 Redes Sociales

Elaborado por: Araceli Dávila M.

Gráfico No. 41 Edición Modulo de Redes Sociales


Módulo: [Editar]
Guardar
Aplicar
Cerrar
Ayuda
Detalles
Tipo de módulo: mod_socialmedialinks
Título: Redes Sociales
Mostrar título: No Sí
Publicado: No Sí
Posición: right
Orden: 0 :: Buscar
Nivel de acceso: Público Registrado Especial
ID: 136
Descripción: <h1>Techheads Social Media v2.0.0</h1>This module displays Web 2.0 style icons that link to your social media profiles. To get started go to the module parameters and choose the icon size you want, then add the URLs of your social media profiles. Any icon/URL fields left empty simply won't be displayed. Be sure to include http:// in all URLs. For support contact us at satechheads@yahoo.com
Asignación de menú
Menús: Todos Ninguna Seleccione un ítem del menú de la lista
Selección de menú:
ExamplePages
Section Blog
Section Table
Category Blog
Category Table
keyconcepts
Extensions
Content Layouts
Example Pages
mainmenu
Parámetros
Parámetros del módulo
Sufijo de clase
Choose Icon Size 128 x 128 px 64 x 64 px 32 x 32 px 16 x 16 px
Align Icons Izquierda Centrar Derecha
Space Between Icons 3px
Tooltip Text
Tooltip Text for RSS Icon RSS
Target New Window Same Window
Include the rel='nofollow' link attribute No Sí
Choose Icon Facebook.png
URL http://www.facebook.co
Choose Icon Twitter 02.png
URL http://www.twitter.com
Choose Icon Hi5.png
URL http://www.hi5.com
Choose Icon Yahoo.png
Internet | Modo protegido: desactivado
100%

Elaborado por: Araceli Dávila M.

Como vemos, este componente tiene la opción de poner una imagen y que esa imagen llame a su URL correspondiente.

2.9.2 VISITANTE

El usuario visitante, será cualquier cibernauta que se encontró en la página con un Mashup de noticias, su gusto a la información, encontró accesos a otros servicios sin necesidad de salir de esta página.

2.9.2.1 Regístrese

Un Usuario puede disfrutar de privilegios al registrarse en el sitio, no podía ser de otra manera, tener un control respectivo de las personas que quieren compartir sus diferentes experiencias con la comunidad Mashup.

Para lo cual damos click en “registro”

Gráfico No. 42 Registro de Usuario nuevo

Elaborado por: Araceli Dávila M.

EL formulario en cuestión es el siguiente:

Gráfico No. 43 Formulario de Registro

Elaborado por: Araceli Dávila M.

Cabe destacar que, el Administrador del sitio dará de alta a los usuarios que cree tener el margen de conducta en el sitio.

Un usuario correctamente loggeado y dado de alta, se muestra asi:

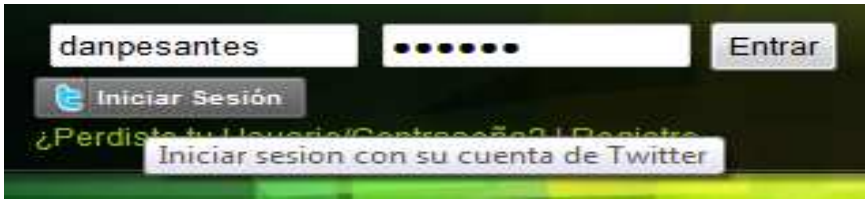
Gráfico No. 44 Usuario Correctamente Loggeado

Elaborado por: Araceli Dávila M.

2.9.2.2 Uso de Twitter

El cibernauta, no necesariamente debe registrarse para hacer uso de los privilegios, puede interactuar con el componente twitter en caso de tener usuario y contraseña.

Gráfico No. 45 Inicio de Sesión Twitter



Elaborado por: Araceli Dávila M.

Luego, el formulario de conexión entre el sitio www.mantentealdia.net y twitter.com, confirma los datos de un usuario válido de Twitter y conecta ambos servicios entre sí.

Queremos aclarar, que esta funcionalidad no depende de nuestros servidores, ya que es exclusiva de los servicios de Twitter, esto quiere decir que si se cae el servicio como tal, no estará disponible en el Mashup.

Gráfico No. 46 Formulario de Conexión mashup-Twitter

Twitter - Windows Internet Explorer
https://oauth.twitter.com/2/authorize?oauth_callback_url=http%3A%2F%2Fww
twitter
¿Eres nuevo en Twitter?
Regístrate ahora
Conecta MANTENTEALDIA con tu cuenta de Twitter para interactuar con tus seguidores y compartir contenidos a través de tu cronología.
¿Ya tienes una cuenta en Twitter? Inicia sesión.
Nombre de usuario o correo electrónico
danpesantes
Contraseña
Conectar
Cancelar
By clicking "Connect" you agree to the Twitter Terms of Service. In particular, some usage information will be shared back with Twitter. For more, see our Privacy Policy. Twitter will never share your password and you can always revoke MANTENTEALDIA's access through your settings page.
Internet | Modo protegido: desactivado
100%

Elaborado por: Araceli Dávila M.

En el momento que la conexión es establecida, queda el usuario loggeado en el servicio normal de www.twitter.com:

Gráfico No. 47 Página principal de Twitter

Elaborado por: Araceli Dávila M.

Y a su vez, es replicada la sesión a nuestro sitio:

Gráfico No. 48 Sesión iniciada, servicio Twitter en Mashup

Elaborado por: Araceli Dávila M.

A su vez, el usuario puede hacer uso del chat entre los usuarios registrados al sitio.

Gráfico No. 49 Chat Habilitado para usuario de Twitter

Elaborado por: Araceli Dávila M.

Y sobre la parte inferior derecha, el usuario puede leer sus tweets, enviados y receptados.

Ver la cantidad de seguidores, e inclusive formatos de sindicación de contenidos.

Gráfico No. 50 Módulo Twitter en Mashup

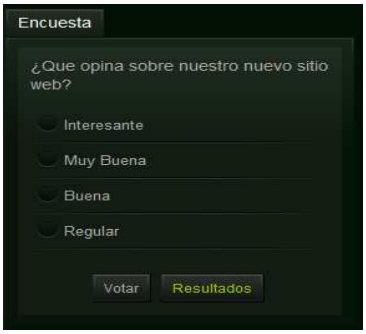
Daniel Pesantes
danpesantes
feed: RSS Feed
follow: Follow updates
updates: 4
followers: 3
following: 5
Following:
Last 3 tweets from danpesantes:
From web
danpesantes: @AntoV25 grande toño todo Ecuador esta contigo esa copa no se va de Inglaterra colocate la bandera de Ecuador en el festejo!!!!
From web
danpesantes: de vuelta al twitter
From web
danpesantes: RT @AlejandroSanz: Entonces que guayaquil? Hay

Elaborado por: Araceli Dávila M

2.9.2.3 Participe en la Encuesta

Todos los visitantes pueden hacer uso de la encuesta, esto para medir el grado de aceptación de un servicio, de un producto. Cabe destacar que el servidor evalúa que una encuesta la puede hacer un usuario dentro del mismo día y de una misma IP.

Gráfico No. 51 Encuestas



Elaborado por: Araceli Dávila M.

2.9.2.4 Chat en vivo

Este servicio, está habilitado entre usuarios de Twitter, correctamente habilitados, y usuarios que se han registrado en nuestro Mashup.

Gráfico No. 52 Chat

Elaborado por: Araceli Dávila M.

2.9.2.5 Chat interactivo

Este servicio ofrece más novedades en cuanto a las forma de chatear ya que contiene dibujos los cuales se puede añadir al momento de chatear, tiene habilitado el sonido, maneja historial del chat para que la persona tenga acceso en el caso de que quiera recordar de manera textual todo lo que ha chateado con la otra persona en ese momento. Tiene la opción de cambiar el color de letra.

Gráfico No. 53 Chat Interactivo

Elaborado por: Araceli Dávila M

2.9.2.6 Infórmese con Noticias desde Noticias Online

Este servicio está conectado a varias páginas de noticias, se actualiza a medida se cambian las noticias en la página principal del diario, de yahoo y de google.

Gráfico No. 54 RSS – Varios RSS de Noticias

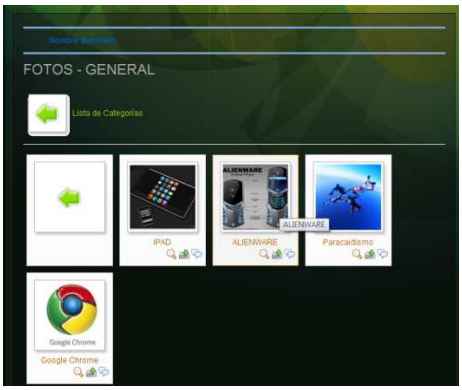
Noticias Online
Tecnología: Google Noticias
InquirerHTC One X, EVO 4G LTE Held at Customs Over ITC RulingTom's GuideIn December of 2011, the International Trade Commission ruled against HTC with regard to two claims made by rival Apple. The result was an import ban on HTC[...]
Leer más...
Tecnología: Google Noticias
The HinduWatch 'Zuckerberg: The Musical!' [VIDEO] MashableFacebook founder Mark Zuckerberg is having the best week ever. He turned 28 on Monday, and his company goes public later this week. The timing couldn't be better for him to be immortalized in song.[...]
Leer más...

Elaborado por: Araceli Dávila M.

2.9.2.7 Suba Fotos, comente acerca de las otras

El servicio de subida de fotos, está disponible a los usuarios respectivamente calificados y habilitados por los Administradores de mashup.

Gráfico No. 55 Fotos



Elaborado por: Araceli Dávila M.

Para subir, una foto se debe escoger la opción respectiva, título de la misma, descripción. Tomar en cuenta el tamaño máximo permitido.

Gráfico No. 56 Subida de Fotos

Elaborado por: Araceli Dávila M.

Y la fotografía se encuentra lista para ser comentada.

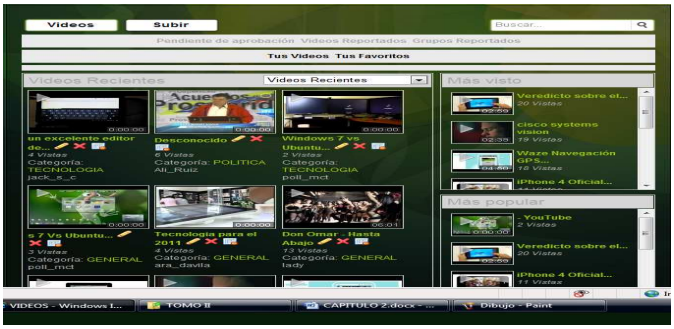
Gráfico No. 57 Comentarios en Fotos

Elaborado por: Araceli Dávila M.

2.9.2.8 Suba un link de Video, comentario de las otras

La sección de Videos, contiene una parte de videos más rankeados (Mas vistos), y los que mas gustan (populares),

Gráfico No. 58 Videos



Elaborado por: Araceli Dávila M.

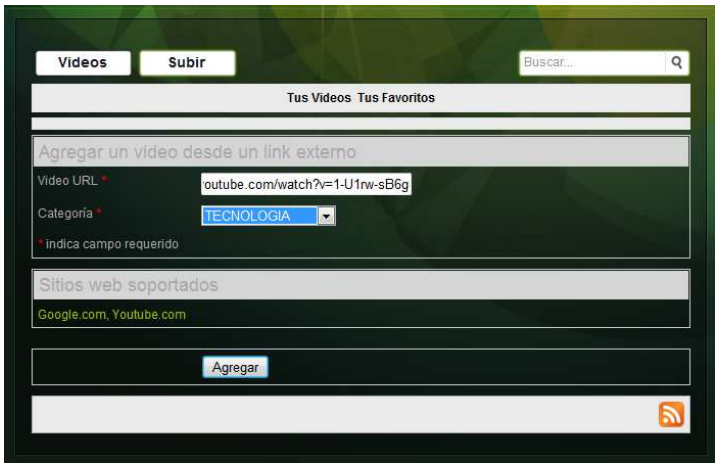
Si queremos ver el video, damos click sobre el que solicitamos, y también queda la posibilidad de comentar dicho video.

Gráfico No. 59 Ver Videos

Elaborado por: Araceli Dávila M.

La forma de subir videos, es solo de un enlace, se tomó esa decisión para no subir archivos pesados a los servidores, sino mas bien se utilzará recursos externos (Youtube, Google)

Gráfico No. 60 Subir Videos

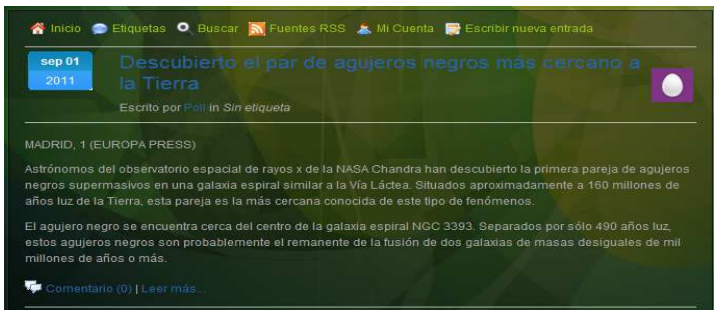


Elaborado por: Araceli Dávila M.

2.9.2.9 Suba su Noticia o artículo de interés

Esta es la entrada principal de noticias de usuarios, donde se puede buscar entre las noticias, debidamente categorizadas y la opción de escribir una nueva entrada, en la parte derecha del título sale el avatar de la persona autora de dicho artículo.

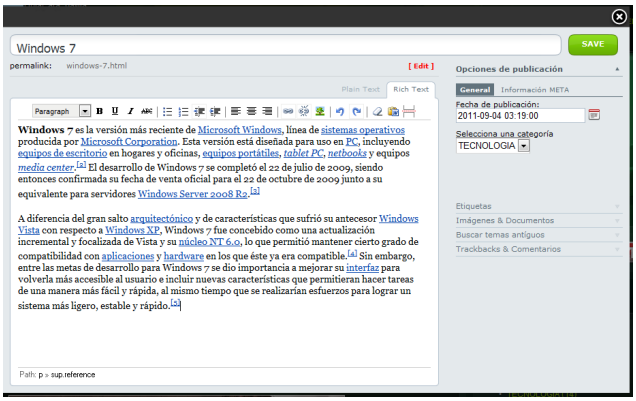
Gráfico No. 61 Noticia escrita por un usuario


Inicio
Etiquetas
Buscar
Fuentes RSS
Mi Cuenta
Escribir nueva entrada
sep 01
2011
Descubierto el par de agujeros negros más cercano a la Tierra
Escrito por Pelli in Sin etiqueta
MADRID, 1 (EUROPA PRESS)
Astrónomos del observatorio espacial de rayos x de la NASA Chandra han descubierto la primera pareja de agujeros negros supermasivos en una galaxia espiral similar a la Vía Láctea. Situados aproximadamente a 160 millones de años luz de la Tierra, esta pareja es la más cercana conocida de este tipo de fenómenos.
El agujero negro se encuentra cerca del centro de la galaxia espiral NGC 3393. Separados por sólo 490 años luz, estos agujeros negros son probablemente el remanente de la fusión de dos galaxias de masas desiguales de mil millones de años o más.
Comentario (0) | Leer más...

Elaborado por: Araceli Dávila M.

En “Escribir nueva entrada”, da apertura a un editor de texto basado en Joomla, para insertar información y dar formato al texto, incluir fotografías e incluso contenido multimedia.

Gráfico No. 62 Editor de texto de Noticias


Windows 7
SAVE
permalink: windows-7.html
[Edit]
Plain Text
Rich Text
Opciones de publicación
General
Información META
Fecha de publicación:
2011-09-04 03:19:00
Selecciona una categoría
TECNOLOGÍA
Etiquetas
Imágenes & Documentos
Buscar temas antiguos
Trackbacks & Comentarios
Windows 7 es la versión más reciente de Microsoft Windows, línea de sistemas operativos producida por Microsoft Corporation. Esta versión está diseñada para uso en PC, incluyendo equipos de escritorio en hogares y oficinas, equipos portátiles, tablet PC, netbooks y equipos media center.[a] El desarrollo de Windows 7 se completó el 22 de julio de 2009, siendo entonces confirmada su fecha de venta oficial para el 22 de octubre de 2009 junto a su equivalente para servidores Windows Server 2008 R2.[3]
A diferencia del gran salto arquitectónico y de características que sufrió su antecesor Windows Vista con respecto a Windows XP, Windows 7 fue concebido como una actualización incremental y focalizada de Vista y su núcleo NT 6.0, lo que permitió mantener cierto grado de compatibilidad con aplicaciones y hardware en los que éste ya era compatible.[4] Sin embargo, entre las metas de desarrollo para Windows 7 se dio importancia a mejorar su interfaz para volverla más accesible al usuario e incluir nuevas características que permitieran hacer tareas de una manera más fácil y rápida, al mismo tiempo que se realizarían esfuerzos para lograr un sistema más ligero, estable y rápido.[5]
Path: p » sup.reference

Elaborado por: Araceli Dávila M.

Cabe mencionar, que este texto puede ser de otras fuentes, el cual muchas veces producto de copiar y pegar, puede traer inclusive el formato html de otro sitio web, lo que deberíamos considerar en revisar periódicamente las noticias, para evitar sorpresas de alteraciones de html del formato de la página nativa Joomla.

2.9.2.10 Los 10 mas Bloggeados

Este es el rankin de las personas que mas noticias han publicado, se puede modificar en la visualización de la cantidad de personas a mostrar.

Gráfico No. 63 Los más Bloggeados

Elaborado por: Araceli Dávila M.

2.9.2.11 Valore un Anuncio Clasificado

En la sección de clasificados, hemos dividido a Servicios y productos, a los tipos de avisos que se pueden escribir. Esta es la sección donde pueden publicar un aviso de su servicio a ofrecer a la clientela.

Gráfico No. 64 Productos y Servicios

CLASIFICADOS
A continuación podra encontrar los distintos productos o servicios que los usuarios están ofertando
Productos
En esta sección usted podrá encontrar todo tipo de accesorios, entre los principales tenemos:
pendrive
hub usb
SD card
monitores
portatiles
celulares
Servicios
En esta sección usted podrá encontrar los distintos servicios que nuestros usuarios han puesto a disposición de usted como son:
soporte técnico
desarrollo de aplicaciones web
desarrollo de aplicaciones moviles
desarrollo de gigantografias

Elaborado por: Araceli Dávila M.

En el caso de Servicios, tenemos un listado de aquellos registrados previamente.

Gráfico No. 65 Lista de servicios disponibles

Servicios
En esta sección usted podrá encontrar los distintos servicios que nuestros usuarios han puesto a disposición de usted como son:
soporte técnico
desarrollo de aplicaciones web
desarrollo de aplicaciones moviles
desarrollo de gigantografias
Filtrar por titulo
Mostrar # 20
Titulo del articulo clics
1 Diseño Web Empresarial 6
Nuevo

Elaborado por: Araceli Dávila M.

En el caso, que el aviso le haya servido a su medida, se debe calificar para obtener una puntuación de los usuarios. El mismo que servirá para medir la valoración del profesional que ofrece sus servicios.

Ya queda a elección del resto de usuarios, hacer uso de un aviso con puntuación baja.

Gráfico No. 66 Valoración del Servicio

Diseño Web Empresarial
Valoración de los usuarios / 0
Pobre El mejor Valoración
Detalle del plan:
• Diseño web: adaptación de una plantilla de Joomla, logo, colores institucionales, menús, etc.
• Total autogestión de contenidos. Modulo de administración del sitio de fácil manejo.
• Hasta 30 páginas web. Contenidos con texto e imágenes
• Imágenes ilimitadas. Las cuales pueden ser incluidas en las páginas del sitio.
• Modulo de estadísticas detalladas (país, navegador, sistema operativo, etc)
• Una animación de introducción (intro) con sonido.
• Dos banner animado dentro del sitio.
• 3 Funcionalidades extras básicas elegida por el cliente
• 2 Funcionalidades extras profesionales elegidas por el cliente.
• 1 formulario de contacto direccionado hacia el correo del administrador del sitio.
• Área de registro de usuarios con verificación de correo.
• Optimización básica para los principales buscadores (Google, Yahoo, Altavista, etc)
• Contador de visitas
• Dominio (nombre de su empresa).com
• 5 Cuentas de email. - por ej: info@(nombre de su empresa).com
• Alta en buscadores y directorios
• 1000 Mb de Hosting Linux

Elaborado por: Araceli Dávila M.

2.9.2.12 Contacte al Administrador del Sitio

En caso que debamos comunicar, algún tema de interés con el Administrador del site, accedemos a esta sección, puede ser para tratar temas de artículos que atenten a propiedad intelectual, artículos que atenten a la integridad de una persona, o

solicitar nuestros servicios para publicar en cualquiera de los temas anteriores, utilizamos el formulario de contacto.

Cabe recalcar, que el correo electrónico llegará a la persona que esta como Administrador del Sitio general, ya que como bien sabemos. El sitio puede tener algunos administradores.

Gráfico No. 67 Formulario de Contacto

Elaborado por: Araceli Dávila M.

2.9.2.13 “Me Gusta”, Funcionalidad de Facebook

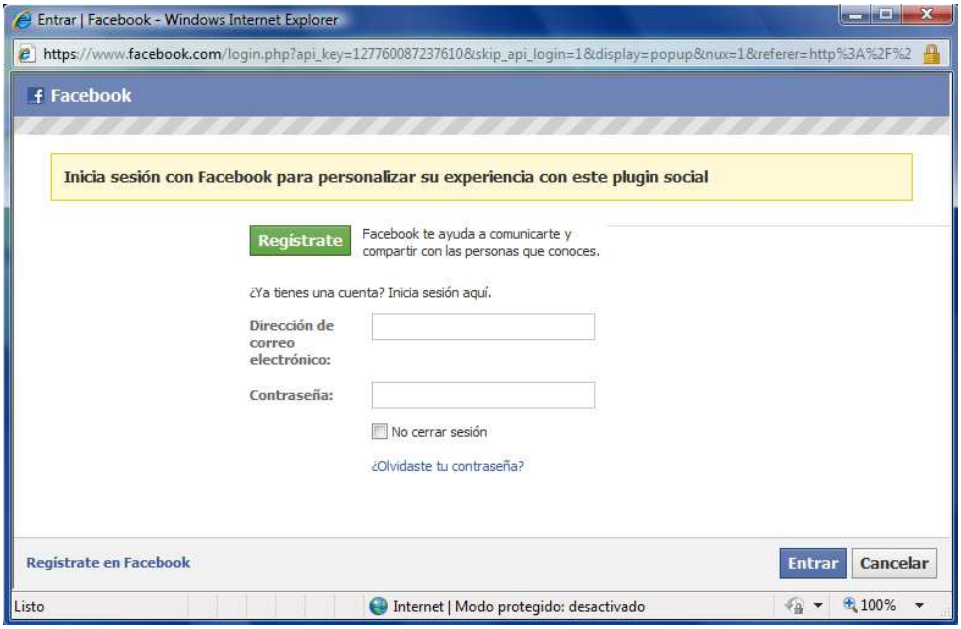
Sobre la página principal del sitio, existe la funcionalidad “Me gusta”, dónde el usuario hace uso de la misma.

Gráfico No. 68 Facebook-“Me Gusta”

Elaborado por: Araceli Dávila M.

Si no está loggeado en Facebook, se abrirá un formulario de acceso rápido a la red social en mención.

Gráfico No. 69 Formulario de Acceso rápido a Facebook



Elaborado por: Araceli Dávila M.

Y automáticamente el contador registra el voto – “Me gusta”

Gráfico No. 70 Voto en “Me gusta” – Facebook

Elaborado por: Araceli Dávila M.

2.9.2.14 “Recomendar”, Funcionalidad Facebook

En el pie de la página, existe la funcionalidad descrita:

Gráfico No. 71 Recomendar

Elaborado por: Araceli Dávila M.

En caso que el usuario no tenga cuenta en Facebook, puede hacer click en “Regístrate” para iniciar una. Esta funcionalidad además presenta a los usuarios que han recomendado el sitio. Para hacer uso de esta funcionalidad, se debe estar previamente loggeado, caso contrario se abre un mini formulario de acceso a Facebook. Y Se registra el voto recomendado (Ver fotografía “Formulario de acceso rápido a Facebook”).

Cabe recalcar que el producto que se recomienda es el uso del sitio www.mantentealdia.net

2.9.2.15 Tweets En Línea

Está disponible el Hashtag **#noticias_mantentealdia**, cualquier mención del mismo será publicado directa y dinámicamente en nuestro sitio.

Gráfico No. 72 Tweets en línea

8
27
Me gusta
Twittear
@mantentealdia_
#noticias_mantentealdia
ara_davila
#noticias_mantentealdia El oro y otros metales preciosos proceden del bombardeo de asteroides enterese en: http://t.co/VDvl89SQ
3 days ago • reply • retweet • favorite
ara_davila
#noticias_mantentealdia Gadafi podría haber escapado a Burkina Faso con un convoy de oro y armas enterese en: http://t.co/VDvl89SQ
3 days ago • reply • retweet • favorite
ara_davila
#noticias_mantentealdia Sigan este sitio para permanecer informado en el portal de noticias mas grande del Ecuador
twitter
Join the conversation

Elaborado por: Araceli Dávila M.

Los usuarios pueden enviar sus tweets desde cualquier plataforma, ya sea móvil o por el sitio oficial. Además por el botón Twittear. Solo deben contener el hashtag en mención.

Gráfico No. 73 Botón Twittear

Elaborado por: Araceli Dávila M.

2.9.2.15 Botón Siguem en Twitter

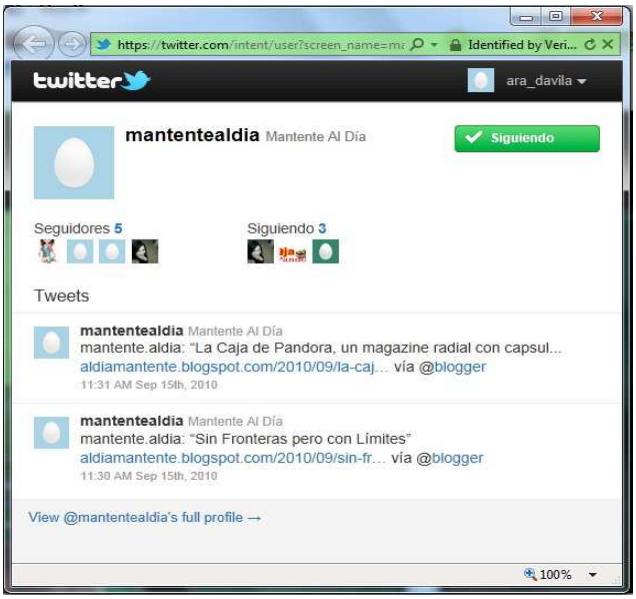
Se encuentra disponible para todas las personas que tengan cuentan en twitter y quieran seguir al sitio mantentealdia.net por medio de la red social de twitter.

Gráfico No. 74 Botón Sigueme de Twitter

Elaborado por: Araceli Dávila M.

Una vez que el usuario le da click en el icono del pájaro el cual es símbolo del twitter automáticamente le presentará otra ventana en la que le podrá seguir y automáticamente seguirá al sitio oficial en twitter el cual se llama mantentealdia.net.

Gráfico No. 75 Siguiendo al sitio en twitter

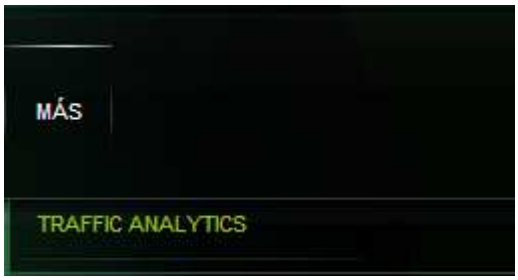


Elaborado por: Araceli Dávila M.

2.9.2.16 Tráfico del Sitio

En la parte principal del sitio mantentealdia.net se encuentra el menú más que al darle click aparecerá otro submenú llamdo Traffic Analytics.

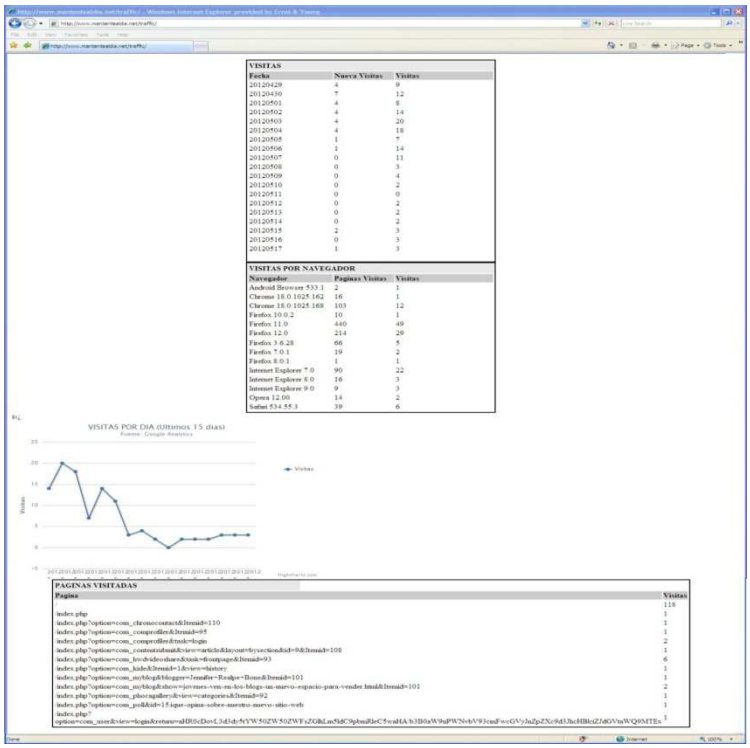
Gráfico No. 76 Menú del Tráfico del Sitio



Elaborado por: Araceli Dávila M.

Cuando el usuario da click en el submenú de Traficc Analytics automáticamente se mostrará una pantalla en la cual presentará el número de personas que han visitado el sitio y en que tipo de explorador han realizado su visit

Gráfico No. 77 Monitoreo de Tráfico del Sitio



Elaborado por: Araceli Dávila M.

2.9.2.17 Módulo adicional de Administración.

Este tipo de administración esta creado para personas que no tenga conocimientos de joomla pero para ingresar al administador deben tener permiso de administrador, por ende cualquier usuario no podrá ser administrador, para lo cual este permiso solo lo otorgará el administrador de joomla el cual maneja todo los permisos y es encargado de monitorear frecuentemente al sitio mantentealdia.net.

Este administrador tiene solo las funciones principales de los módulos de administración de joomla como crear y eliminar usuario, ingresar noticias, crear categorías entre otros y fue realizado específicamente para personas que no tengan el conocimiento de joomla y que recién empiecen en el ámbito de administración de sitios web.

Este administrador se lo puede encontrar en el menú principal del sitio www.mantentealdia.net que se llama mi cuenta que al darle click aparecerá un submenú llamado administración.

Gráfico No. 78 Menú para ingresar al administrador

Elaborado por: Araceli Dávila M.

Cuando el usuario da click en el submenú de Administración automáticamente se mostrará una pantalla en la cual presentará el administrador del sitio, el cual lo deberá manejar con la responsabilidad que amerita dicha función ya que tendrá acceso y permisos especiales para monitorear el sitio www.mantenteladia.net

Gráfico No. 79 Monitoreo de Administración

Elaborado por: Araceli Dávila M.